NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

KALMAN FILTERING APPROACH
TO
BLIND EQUALIZATION

by

Mehmet Kutlu

December, 1993

Thesis Advisor : Roberto Cristi
Second Reader : Phillip E. Pace

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Master's Thesis

4. TITLE AND SUBTITLE KALMAN FILTERING APPROACH TO BLIND EQUALIZATION

5. FUNDING NUMBERS

6. AUTHOR(S) Mehmet Kutlu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

A

13. ABSTRACT *(maximum 200 words)*

Digital communication systems suffer from the channel distortion problem which introduces errors due to intersymbol interference. The solution to this problem is provided by equalizers which use a training sequence to adapt to the channel. However in many cases in which a training sequence is unfeasible, the channel must be adapted blindly. Most of the blind equalization algorithms known so far have problems of convergence to local minima. Our intention is to offer an alternative approach by using extended Kalman filtering and hidden Markov models. They seem to yield more efficient algorithms which take the statistics of the transmitted sequence into consideration. The theoretical development of these new algorithms is discussed in this thesis. Also these algorithms have been simulated under different conditions. The results of simulations and comparisons with existing systems are provided. The models for simulations are presented as MATLAB codes.

14. SUBJECT TERMS
Communications, Digital Signal Processing, Equalization, Kalman Filters, Markov Models

15. NUMBER OF PAGES
80

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

Kalman Filtering Approach
to
Blind Equalization

by

Mehmet Kutlu
Lieutenant Junior Grade, Turkish Navy
B.S.E.E., Naval Postgraduate School, 1993

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1993

Author: Mehmet Kutlu

Approved by: Roberto Cristi, Thesis Advisor

Phillip E. Pace, Second Reader

Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

Digital communication systems suffer from the channel distortion problem which introduces errors due to intersymbol interference. The solution to this problem is provided by equalizers which use a training sequence to adapt to the channel. However in many cases in which a training sequence is unfeasible, the channel must be adapted blindly. Most of the blind equalization algorithms known so far have problems of convergence to local minima. Our intention is to offer an alternative approach by using extended Kalman filtering and hidden Markov models. They seem to yield more efficient algorithms which take the statistics of the transmitted sequence into consideration. The theoretical development of these new algorithms is discussed in this thesis. Also these algorithms have been simulated under different conditions. The results of simulations and comparisons with existing systems are provided. The models for simulations are presented as MATLAB codes.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

In appreciation for their time, effort, and patience, many thanks go to my instructors, advisors, and the staff and faculty of the Electrical and Computer Engineering Department, NPS.

I would like to offer special thanks to my family for their patience, support and encouragement.

I. INTRODUCTION

One of the major practical problems in digital communication systems is channel distortion which causes errors due to intersymbol interference. Since the source signal is in general broadband, the various frequency components experience different steady state amplitude and phase changes as they pass through the channel, causing distortion in the received message. This distortion translates into errors in the received sequence.

Our problem as communication engineers is to restore the transmitted sequence or, equivalently, to identify the inverse of the channel, given the observed sequence at the channel output. This task is accomplished by adaptive equalizers.

Typically, adaptive equalizers used in digital communications require an initial training period, during which a known data sequence is transmitted. A replica of this sequence is made available at the receiver in proper synchronism with the transmitter, thereby making it possible for adjustments to be made to the equalizer coefficients in accordance with the adaptive filtering algorithm employed in the equalizer design. When the training is completed, the equalizer is switched to its decision directed mode.

However, there are practical situations where it would be highly desirable for a receiver to be able to achieve complete adaptation without the cooperation of the transmitter. This can be the case of an unfriendly receiver. Also in the communication schemes where the channel parameters change with time, such as in mobile communication systems, a training sequence must be repeated, leading to waste in the

channel utilization. In these cases, the equalization must be performed without a training sequence. In other words, the receiver is blind to the specific transmitted source sequence.

The development of a parameter estimation algorithm appropriate for a blind equalizer adaptation is hindered by the lack of the reference signal by most recursive schemes. If the source sequence values were known to be independent and identically distributed (i.e., i.i.d.) this property could be used to restore the output of the channel by filters designed via estimation theory techniques. The presumption is that proper deconvolution of the received signal would restore the source's transmitted sequence. To provide the reader with a better understanding of our approach to the blind equalization problem, we will review below the available blind equalization algorithms and the problems with them.

A. AVAILABLE ALGORITHMS

1. Constant Modulus Adaptive Algorithm (CMA)

In the Constant Modulus Algorithm (CMA), the error between the magnitude (modulus) of the equalizer output and a constant is recursively minimized, resulting in an algorithm of complexity similar to the Least Mean Square (LMS). The motivation for these methods is that by restoring the modulus of the received signal, the channel impulse response should be implicitly estimated and intersymbol interference removed. Since the magnitude of the equalizer output is independent of the absolute phase, the cost

functions in these algorithms are independent of the transmitted data sequence, and hence, they are capable of blind deconvolving the received sequence.

The baseband model of a digital communication system consists of the cascade connection of a linear communication channel and a blind equalizer, depicted in Figure (1.1).

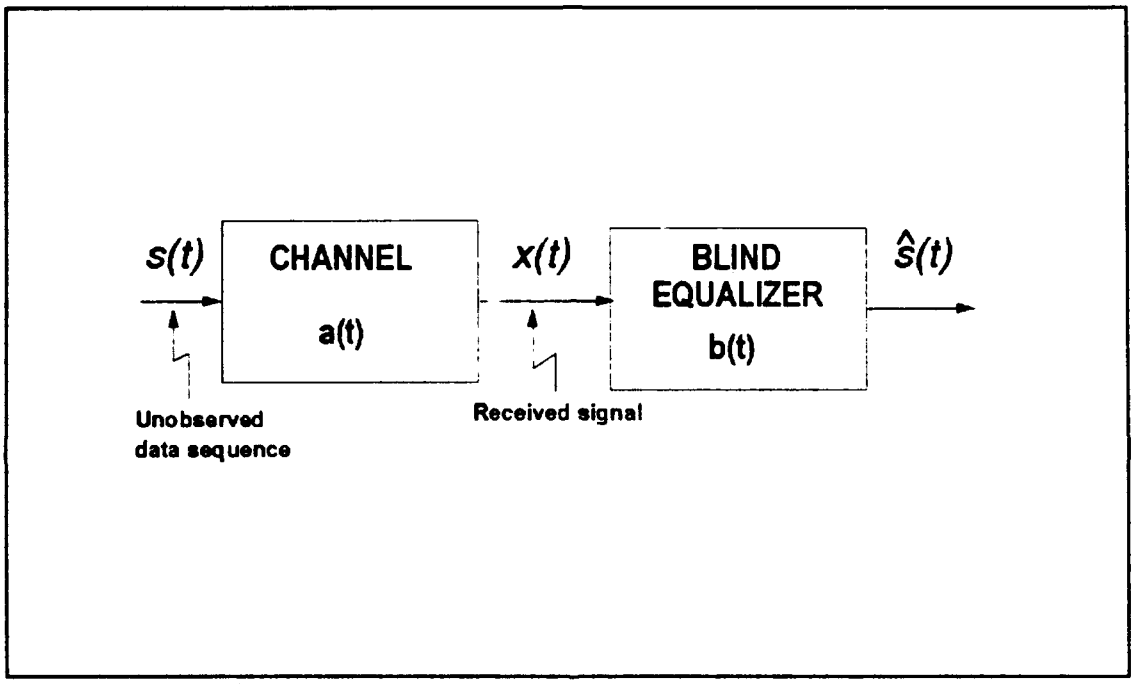

Figure 1.1 Model of Digital Communication System

The channel includes the combined effects of a filter at the transmitter, a transmission medium, and a filter at the receiver. If we assume it to be linear and time invariant or slowly varying, it is characterized by an unknown impulse response $a(t)$. The nature of the impulse response $a(t)$ is determined by the type of the modulation employed. We may thus describe the input-output relation of the channel by the convolution sum

$$x(t) = \sum_i a_i(t)\ s(t-i) \qquad \forall\ t \tag{1.1}$$

where $s(t)$ is the data sequence applied to the channel input and $x(t)$ is the resulting channel output. We assume that

$$\sum_i a_i^2(t) = 1\ . \tag{1.2}$$

This implies the use of an automatic gain control (AGC) that keeps the variance of the channel output $x(t)$ constant.

Let $b(t)$ denote the impulse response of the ideal inverse filter, which is related to the impulse response $a(t)$ of the channel as follows

$$\sum_k b_k(t)\ a_{m-k}(t) = \delta_m \tag{1.3}$$

where δ_m is the Kronecker delta.

$$\delta_m = \begin{cases} 1 & if \quad m=0 \\ 0 & if \quad m\neq 0 \end{cases} \tag{1.4}$$

An inverse filter defined in this way is "ideal" in the sense that it reconstructs the transmitted data sequence $s(t)$ correctly.

For the situation described herein, the impulse response $a(t)$ is unknown and therefore we cannot use Equation (1.3) to determine the inverse filter. Instead we use an iterative deconvolution procedure to complete an approximate inverse filter characterized by $\hat{b}_i(t)$. The index i refers to the tap-weight number in the transversal

filter realization of the approximate filter [Ref.1]. Thus at the nth iteration we have an approximately deconvolved sequence.

$$y(t) = \sum_{i=-L}^{L} \hat{b}_i(t)\ x(t-i) \qquad (1.5)$$

The convolution sum for the ideal inverse filter is infinite in extent in that the index i ranges from $-\infty$ to ∞. So we may rewrite Equation (1.5) as follows;

$$y(t) = \sum_{i} \hat{b}_i(t)\ x(t-i), \qquad \hat{b}_i(t) = 0 \quad for \quad |i| > L$$

or, equivalently

$$y(t) = \sum_{i} b_i(t)\ x(t-i) + \sum_{i} [\hat{b}_i(t) - b_i]\ x(t-i)\ . \qquad (1.6)$$

If we let

$$v(t) = \sum_{i} [\hat{b}_i(t) - b_i]\ x(t-i), \qquad \hat{w}_i = 0 \quad for \quad |i| > L \qquad (1.7)$$

then we can write

$$y(t) = x(t) + v(t) \qquad (1.8)$$

where the term $v(t)$ is called "convolutional noise", representing the residual intersymbol interference that results from the use of an approximate inverse filter.

The inverse filter output $y(t)$ is next applied to a zero memory nonlinear estimator producing estimate $\hat{s}(t)$ for the datum $s(t)$ [Ref.1]. We may thus write

$$\hat{s}(t) = g[y(t)] \ . \qquad (1.9)$$

Ordinarily, we find that the estimate $\hat{s}(t)$ is not reliable enough. Nevertheless, we may use it in an adaptive scheme to obtain a better estimate at iteration $t+1$. We have a variety of linear adaptive filtering algorithms to perform this estimation.

By looking at the problem, we note the following:

- The ith tap input at the transversal filter at iteration t is $x(t-i)$.
- Viewing the nonlinear estimate $\hat{s}(t)$ as the desired response, and recognizing that the corresponding transversal filter output is $y(t)$, we may express the estimation error for the iterative deconvolution procedure as

$$e(t) = \hat{s}(t) - y(t) \ . \qquad (1.10)$$

- The ith tap weight $\hat{b}_i(t)$ at iteration t represents an old parameter estimate.

Accordingly, the updated value of ith tap weight at iteration $t+1$ is computed as follows

$$\hat{b}_i(t+1) = \hat{b}_i(t) + \mu\, x(t-i)\, e(t), \quad \forall\, i \qquad (1.11)$$

where μ is the step size parameter. Equations (1.5),(1.9),(1.10) and (1.11) constitute the iterative deconvolution algorithm, known as CMA, for the blind equalization of a real baseband channel. The ensemble averaged cost function corresponding to the tap weight update Equation (1.11) is defined by

$$
\begin{aligned}
J(t) &= E\,[\,e^2(t)\,] \\
&= E\,[\,(\hat{s}(t)-y(t))^2\,] \\
&= E\,[\,(g(y(t))-y(t))^2\,]\ .
\end{aligned}
\tag{1.12}
$$

2. Decision-Directed Algorithm

Figure (1.2) presents a block diagram of the equalizer using the decision-directed algorithm. The main difference between this algorithm and the CMA is the type of zero memory nonlinearity imbedded in the equalizer. Specifically, the conditional mean estimation of the blind equalizer is replaced by a threshold decision device, i.e.,

$$
\hat{s}(t) = dec[y(t)]\ . \tag{1.13}
$$

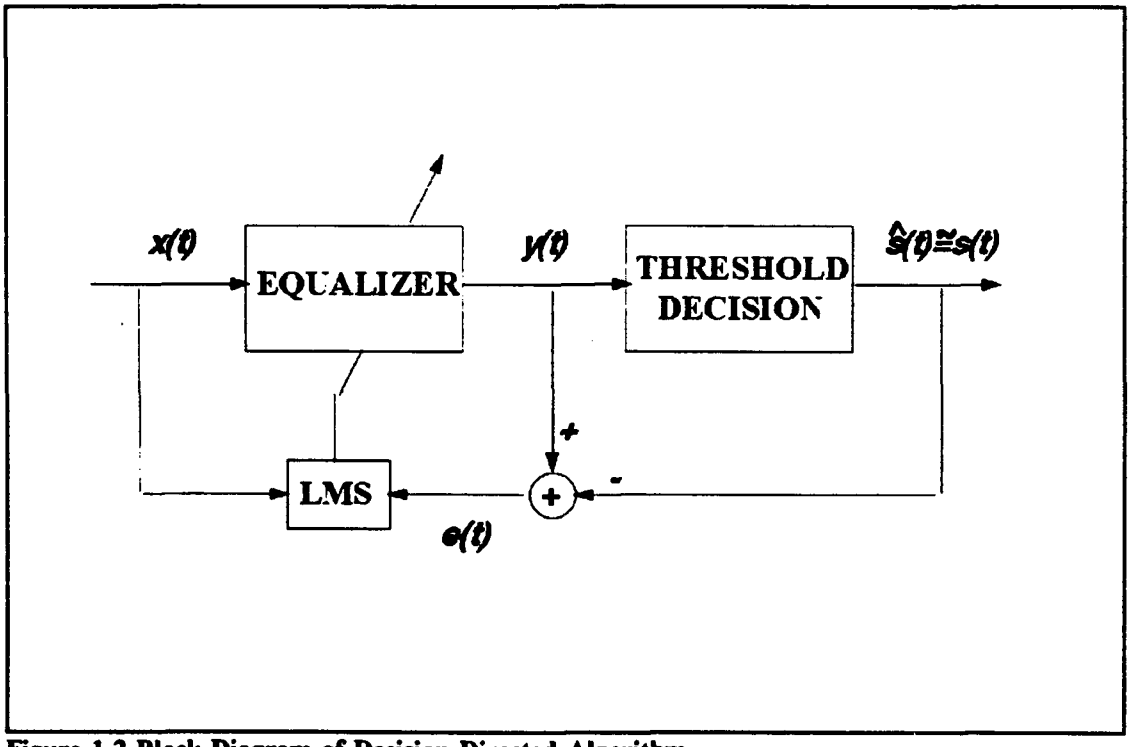

Figure 1.2 Block Diagram of Decision Directed Algorithm

For Binary Phase Shift Keying (BPSK) we may write

$$s(t)=\begin{cases} +1 & \text{for symbol 1} \\ -1 & \text{for symbol 0} \end{cases} \tag{1.14}$$

so

$$dec[y(t)]=sgn[y(t)] \tag{1.15}$$

where $sgn(.)$ is the signum function.

The Sato and Godard algorithms [Ref.1], which are generally used in practice, are special cases of the CMA and decision-directed algorithms. For the Sato Algorithm the zero memory nonlinear function can be shown to be

$$\hat{s}(t) = \gamma\ sgn[y(t)] \tag{1.16}$$

where

$$\gamma = \frac{E[s^2(t)]}{E[|s(t)|]} \ . \tag{1.17}$$

For the Godard algorithm the zero memory nonlinear function is defined as

$$\hat{s}(t) = \frac{y(t)}{|y(t)|}\left[|y(t)| + R_p\ |y(t)|^{p-1} - |y(t)|^{2p-1}\right] \tag{1.18}$$

where p is a positive integer, and R_p is a positive real constant defined by

$$R_p = \frac{E[|s(t)|^{2p}]}{E[|s(t)|^{p}]} \ . \tag{1.19}$$

3. Higher-Order Moments

A different approach is to use higher order moments. Just to give the general idea, we know that the data sequence $s(t)$ transmitted through the channel is non-gaussian. But the output sequence $x(t)$,

$$x(t) = \sum_i a_i(t)\ s(t-i) \qquad \forall\ t$$

defined in Equation (1.1), tends to be Gaussian if the length of the impulse response is long enough (or at least more Gaussian than $s(t)$). It turns out that the Gaussianity of a sequence can be measured by the Kurtosis $K(x(t))$ associated with $x(t)$ [Ref.2], defined as

$$K(x(t)) = E\{|x(t)|^4\} - 2\ E^2\{|x(t)|^2\} - |E\{x(t)^2\}|^2\ , \tag{1.23}$$

which uses the higher order moments of $x(t)$. It can be shown that $K(x(t)) = 0$ if and only if $x(t)$ is Gaussian. We can construct an algorithm by using the Kurtosis where the impulse response of the equalizer $b(t)$ is such that the filtered signal $\hat{s}(t)$ has a maximum Kurtosis.

$$\hat{s}(t) = \sum_i b_i(t)\ x(t-i) \tag{1.24}$$

The maximization is very non-linear, and more information can be found in [Ref.2].

B. PROBLEMS WITH THE EXISTING ALGORITHMS

1. Convergence to Local Minima

One of the major problems in the algorithms based on CMA, is convergence to local minima. The ensemble averaged cost function is defined by Equation (1.12),

$$J(t)=E\ [\ (g(y(t))-y(t))^2]$$

where $y(t)$ denotes the received sequence defined by Equation (1.5). In the linear case of the LMS algorithm, the cost function is a quadratic (convex) function of the tap weights and therefore has a unique minimum. By contrast, the cost function $J(t)$ of Equation (1.12) is a nonconvex function of the tap weights. This means that, in general, the error performance surface of the iterative deconvolution procedure described herein may have more than one local minimum. The nonconvexity of the cost function $J(t)$ arises from the fact that the estimate $\hat{s}(t)$, is produced by passing the linear combiner output $y(t)$ through a zero memory nonlinearity, and also because of $y(t)$ itself a function of the tap weights.

Therefore, nonoptimum fit of the convergent equalizer to the channel inverse occurs, which lead us to low performance of the overall system. Many case studies have been shown in [Ref.3]. Different algorithms have been tried, error functions have been derived and the results have been shown as plots.

Also, if the channel drifts considerably, any decision-directed algorithm looses track of the channel. So, their usage is limited in varying channel conditions.

In this thesis we address the problem of applying optimal estimation techniques to the blind equalization problem. In Chapter II we introduce an algorithm that uses extended Kalman filtering, in Chapter III we discuss an alternate approach which requires a bank of parallel Kalman filters, and in Chapter IV we consider a new algorithm by using the hidden Markov models as well as the extended Kalman filtering, that combines the equalizer with the decoder.

II. CMA LIKE BLIND EQUALIZATION ALGORITHMS BY USING EXTENDED KALMAN FILTERING

As we have discussed in the previous chapter blind equalization is an iterative process mostly based on the optimal performance of prediction-error-based recursive identifiers. Since local convergence and channel drifting constitute a problem to be addressed; the effectiveness of the algorithms based on CMA, which are known so far, is still object of discussion. These facts lead us to consider a class of blind equalization algorithms based on a Kalman filtering approach. One of the approaches is essentially a variation of the CMA algorithm which has been discussed previously. Since the Kalman filter is used for a linear model of the signal, the use of the extended Kalman filter which takes nonlinearities into account, and can still be used effectively in nonlinear and linear models, is considered. By this approach, we hope to address the local convergence problem, which is the main concern for the CMA algorithms. Before presenting our approach we will briefly describe the derivation of the Kalman and Extended Kalman Filter equations, in order to help to understand the main idea behind the new algorithm.

A. AN OUTLINE OF THE USE OF KALMAN FILTERS FOR STATE ESTIMATION (DISCRETE TIME CASE)

1. Derivation of the Kalman Filter Equations

We consider the estimation of the states of a system as represented by a system dynamics model of the form

$$x(t+1) = \mathbf{A}_t x(t) + \mathbf{B}_t s(t) + w(t) \tag{2.1}$$

$$y(t) = \mathbf{C}_t x(t) + v(t) \tag{2.2}$$

in which $w(t)$ represents a zero mean white noise disturbance intensity, and $v(t)$ is a zero mean white noise corrupting the sensor measurement signals, $y(t)$. Their autocorrelations are given by

$$\begin{aligned} \mathbf{E}[w(t)]\,;\mathbf{E}[w(t)\,w^T(t)] &= \mathbf{Q}_t\ \delta(m) \\ \mathbf{E}[v(t)]\,;\mathbf{E}[v(t)\,v^T(t)] &= \mathbf{R}_t\ \delta(m) \end{aligned} \tag{2.3}$$

where $\mathbf{R}_t$ and $\mathbf{Q}_t$ are positive definite matrices and $\delta(m)$ is the discrete-time impulse function.

$$\delta(m) = \begin{cases} 1 & if \quad m=0 \\ 0 & if \quad m\neq 0 \end{cases} \tag{2.4}$$

The term $s(t)$ represents the system input, also $\mathbf{A}_t$, $\mathbf{B}_t$ and $\mathbf{C}_t$ are matrices of appropriate dimension and possibly time varying.

To develop a filter that is recursive for real time applications we define the following algorithm.

STEP 1 : Prediction of the state estimate based on the model and the corresponding new measurement, $\hat{y}(t)$.

$$\hat{x}(t+1|t) = A_t \hat{x}(t|t) + B_t s(t) \tag{2.5}$$

STEP 2 : Update the estimation errors covariance matrix, P_t.

$$P_t(t+1|t) = A_t P_t(t|t) A_t^T + Q_t \tag{2.6}$$

STEP 3 : Recompute the correction (Kalman) gain, K_t.

$$K_t(t+1) = P_t(t+1|t) C_t^T [C_t P_t(t+1|t) C_t^T + R_t]^{-1} \tag{2.7}$$

Here we can see that K_t is determined by A_t, C_t, Q_t and R_t only, not by the measurement, $y(t)$.

STEP 4 : Correction of estimate, based on the error between the new measurement $y(t+1)$ and its prediction, at time $t+1$.

$$\hat{y}(t+1|k) = C_t \hat{x}(t+1|t) \tag{2.8}$$

$$\hat{x}(t+1|t+1) = \hat{x}(t+1|t) + K_t(t+1)\,[y(t+1) - \hat{y}(t+1|t)] \tag{2.9}$$

STEP 5 : Correct the estimation errors covariance matrix, P_t.

$$P_t(t+1|t+1) = [I - K_t(t+1)\, C_t]\, P_t(t+1|t) \tag{2.10}$$

Also initial conditions must be chosen carefully where

$$\hat{x}(0|0) = \mathbf{E}[x(0)] \tag{2.11}$$

is the best estimate at $t=0$ and,

$$P_0 = cov[x(0)] = \mathbf{E}[\{x(0) - \hat{x}(0|0)\}\{x(0) - \hat{x}(0|0)\}^T] = \sigma_0^2 I \tag{2.12}$$

is a positive definite matrix generally in diagonal form. It shows the confidence on initial conditions.

2. The Extended Kalman Filter Equations

The Kalman Filter was derived for the case of linear state and measurement equations. This filter can be generalized to operate for nonlinear state equations and measurement equations in a simple manner. This generalization will be referred to as the Extended Kalman Filter.

A general non-linear system is in the form

$$x(t+1) = f(x(t), s(t), w(t), t)$$
$$y(t+1) = f(x(t), v(t), t)$$

where all the variables are consistent with the linear case. Then,

$$\mathbf{E}[w(t)]\ ; \mathbf{E}[w(t)\, w^T(t)] = Q_t\ \delta(m)$$
$$\mathbf{E}[v(t)]\ ; \mathbf{E}[v(t)\, v^T(t)] = R_t\ \delta(m)$$

are also valid for this case. The Kalman filtering algorithm can be modified to account for the nonlinearities as follows:

STEP 1 :

$$\hat{x}(t+1|t) = f(\hat{x}(t|t), s(t), 0, t) \tag{2.13}$$

STEP 2 :

$$P_t(t+1|t) = \hat{A}_t P_t(t|t) \hat{A}_t^T + Q_t \tag{2.14}$$

where

$$\hat{A}_t = \frac{\partial f(x(t), s(t), w(t), t)}{\partial x(t)} \Big|_{\substack{x(t)=\hat{x}(t|t) \\ s(t)=s(t) \\ w(t)=0}} \tag{2.15}$$

STEP 3 :

$$K_t(t+1) = P_t(t+1|t) \hat{C}_t^T [\hat{C}_t P_t(t+1|t) \hat{C}_t^T + \hat{D}_t R_t \hat{D}_t^T]^{-1} \tag{2.16}$$

where

$$\hat{C}_t = \frac{\partial g(x(t), v(t), t)}{\partial x(t)} \Big|_{\substack{x(t)=\hat{x}(t|t) \\ v(t)=0}} \tag{2.17}$$

and

$$\hat{D}_t = \frac{\partial g(x(t), v(t), t)}{\partial v(t)} \Big|_{\substack{x(t)=\hat{x}(t|t) \\ v(t)=0}} \tag{2.18}$$

STEP 4 :

$$\hat{y}(t+1|k) = g(\hat{x}(t+1|t), 0, t) \tag{2.19}$$

$$\hat{x}(t+1|t+1) = \hat{x}(t+1|t) + K_t(t+1) [y(t+1) - \hat{y}(t+1|t)]$$

STEP 5 :

$$P_t(t+1|t+1) = [I - K_t(t+1)\,\hat{C}_t]\,P_t(t+1|t) \qquad (2.20)$$

for the initial conditions

$$\hat{x}(0|0) = \mathbf{E}[x(0)]$$

and

$$P_0 = cov[x(0)] \ .$$

B. SOLUTION TO BLIND EQUALIZATION PROBLEM BY KALMAN FILTERING

In this section we show how the extended Kalman filtering can be applied for blind equalization. Let X_t be the received signal, and W_t be the weights of the equalizer. The purpose of the equalizer is to restore the original information sequence, i.e.,

$$Y_t = \hat{S}_t = S_t$$

For a binary phase shift keying (BPSK) signal $S_t \in \{+1,-1\}$ and the optimal equalization is achieved when $|y_t|^2 = 1$. Assuming the weights constant in time, we can write a state space equation for the observations as

$$\boldsymbol{W}_t(t+1) = \boldsymbol{W}_t(t) \tag{2.21}$$

$$|y_t|^2 = \boldsymbol{W}_t^T(t)\,\boldsymbol{X}_t(t)\,\boldsymbol{X}_t^T(t)\,\boldsymbol{W}_t(t) \quad . \tag{2.22}$$

As we can see, Equations (2.21) and (2.22) are similar to Equations (2.1) and (2.2) with added nonlinearity.

By applying the extended Kalman filter approach to the blind equalization, the new algorithm becomes as follows:

Initial Conditions:

$$\hat{x}(0|0) = \mathbf{E}[x(0)]$$

$$P_0 = cov[x(0)] = \sigma_0^2 I$$

STEP 1 :

$$\hat{W}(t+1|k) = \hat{W}(t) \tag{2.23}$$

STEP 2 :

$$\hat{\boldsymbol{P}}_t(t+1|t) = I\hat{\boldsymbol{P}}_t(t|t)\,I^T \tag{2.24}$$

since

$$\hat{\boldsymbol{A}}_t = \frac{\partial f(w(t),0,0,t)}{\partial w(t)} = I \tag{2.25}$$

STEP 3 :

$$\boldsymbol{K}_t(t+1) = \boldsymbol{P}_t(t+1|t)\,\hat{\boldsymbol{C}}_t^T[\hat{\boldsymbol{C}}_t\boldsymbol{P}_t(t+1|t)\,\hat{\boldsymbol{C}}_t^T+\boldsymbol{d}] \tag{2.26}$$

where

$$\hat{\boldsymbol{C}}_t=\frac{\partial g(w(t),0,t)}{\partial w(t)} = 2\,[w_t(t)]^T[x_t(t)]\,[x_t(t)]^T \tag{2.27}$$

$$\hat{\boldsymbol{D}}_t=\frac{\partial g(w(t),0,t)}{\partial v(t)} = 0 \tag{2.28}$$

A small error is needed for the Kalman filter in order to work properly, so a scaler value d is added to the Kalman gain calculation at this step.

STEP 4 :

$$\hat{\boldsymbol{Y}}_t^2(t+1|t) = \boldsymbol{W}_t^T(t)\,\boldsymbol{X}_t(t)\,\boldsymbol{X}_t^T(t)\,\boldsymbol{W}_t(t) \tag{2.29}$$

$$\hat{W}_t(t+1|t+1) = \hat{W}_t(t+1|t) + \boldsymbol{K}_t(t+1)\,[1-\hat{y}_t(t+1|t)^2] \tag{2.30}$$

STEP 5 :

$$\boldsymbol{P}_t(t+1|t+1) = [I-\boldsymbol{K}_t(t+1)\,\hat{\boldsymbol{C}}_t]\,\boldsymbol{P}_t(t+1|t) \tag{2.31}$$

C. FEASIBILITY OF THE BLIND EQUALIZATION BY KALMAN FILTERING

As we stated earlier, an admissible source-channel-equalizer combination need not always result in an optimum fit of the convergent equalizer to the channel inverse. So,

we need to study the average behavior of the Kalman Filtering Equalizer algorithm. In this chapter we will consider only linear, time-invariant, stable, channel models and binary real sources, as in [Ref.3]. The same steps have been followed, in order to make a reasonable comparison for our new algorithm.

Figure (2.1) is an illustration of the performance with various fixed equalizer parametrizations. A zero-mean, binary input sequence $s(t)$ is applied to a channel. The received signal $x(t)$ is passed through the equalizer. To provide a sense of "recent average" performance, the sequence of instantaneous squared recovery errors between s and $\hat{s}$ is observed. An average squared error of zero or four denotes perfect performance for a differentially encoded signal and an average squared error greater than 0.4 but less than 3.6 indicates a practically unacceptable error rate of over 10%.

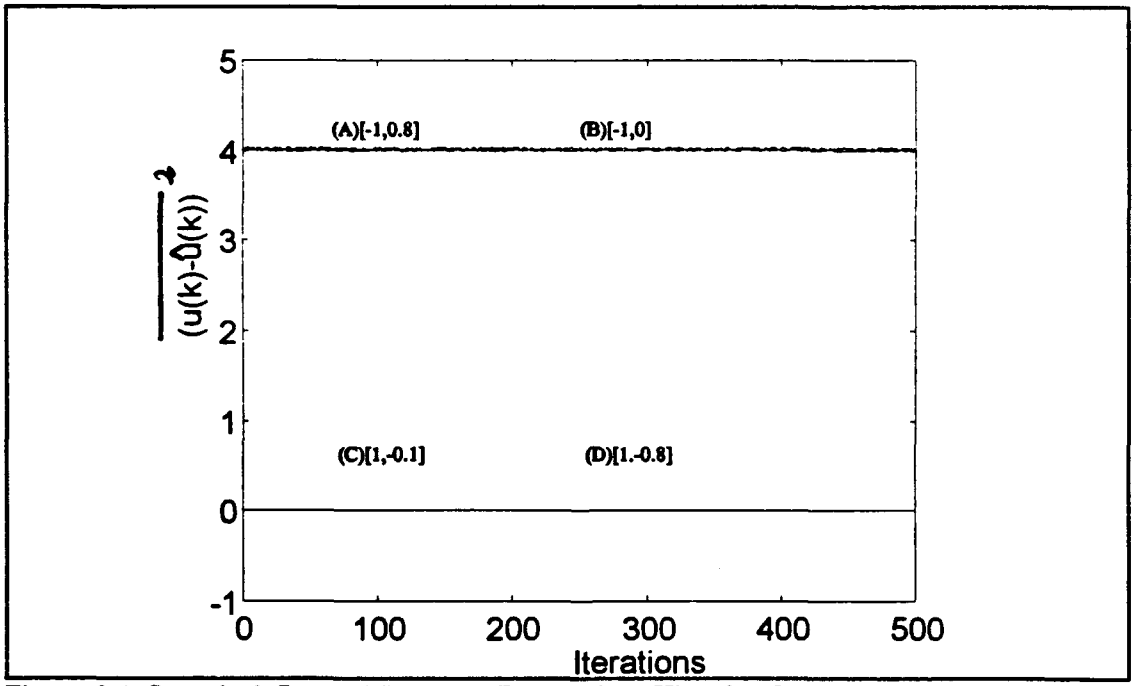

Figure 2.1 Smoothed Squared-Recovery-Error Time Histories for Different Fixed Equalizer Parametrizations [(b_0 , b_1)]

In Figure (2.1) all (b_0,b_1) settings (1,-0.8), (-1,0.8), (-1,0) and (1,-0.1) result in perfect performance . The squared recovery errors associated with (1,-0.8) and (1,-0.1) are perfectly zeroed. The squared recovery error of 4 associated with (-1,0.8) and (-1,0), helps provide the exact negative of s as $\dot{s}$ at each sample instant. If this figure is compared with [Ref.3:Fig.3], it can be seen that the exponential rise does not appear for the Kalman filtering algorithm since we did not use a smoothing filter. Also a (b_0,b_1) setting of (0,1) can not be seen in Figure (2.1) since in this case the model is undetermined.

As a second step, the performance of the Kalman filtering algorithm for different initializations is studied. The (b_0,b_1) setting of (1,-0.6) is used for the equalizer then the initial values on a circle of radius 2 is applied to the system. The results are shown in Figure (2.2). If compared to [Ref.3:Fig.5 and 6] the Kalman filter does not exhibit the

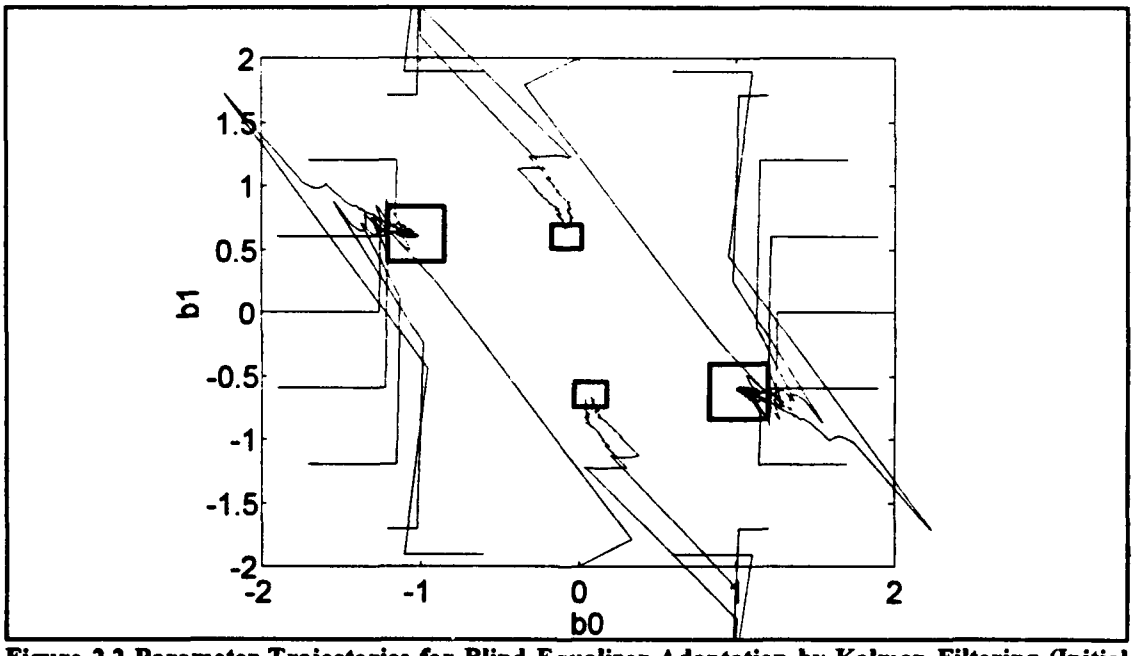

Figure 2.2 Parameter Trajectories for Blind Equalizer Adaptation by Kalman Filtering (Initial Values on the Circle of Radius 2,Final Values in Black Boxes,Local Minima Are Small)

amount of local minima shown by other CMA algorithms. For most of the initial conditions the parameter vector converges to the global minima.

For a perfect solution to the problem, we need to understand the behavior of the algorithm when convergence to local minima occurs. If we initialize (b_0,b_1) at (0,2) the algorithm converges to a global minimum. The normalized squared recovery errors between s and $\hat{s}$ for this case show a distribution equivalent to N(0,1). The situation is shown in Figure (2.3).

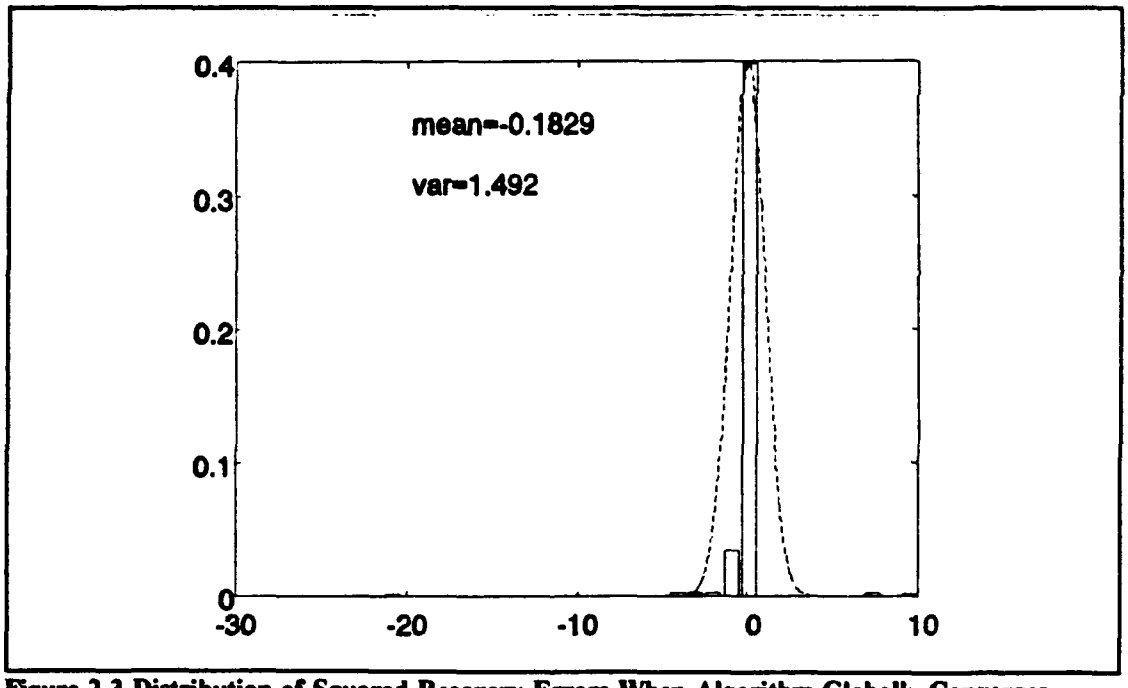

Figure 2.3 Distribution of Squared Recovery Errors When Algorithm Globally Converges

Also, we can see from Figure (2.2) that there are initial conditions which lead to local minima. As in the example of initialization (0.6,-1.9), the algorithm converges to local minimum. In this case, the normalized squared error between s and $\hat{s}$ has a larger mean and variance than the first case. Also, it does not show a normal situation. The

situation is shown in Figure (2.4). This property can be used to determine the convergence to a local minimum.

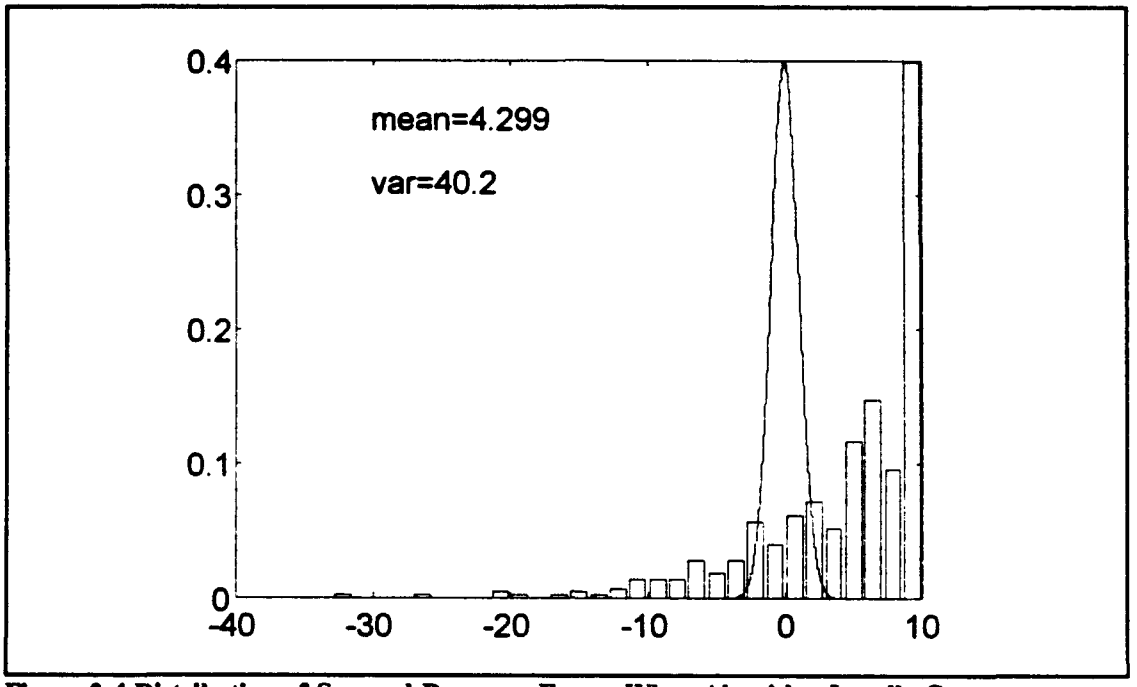

Figure 2.4 Distribution of Squared Recovery Errors When Algorithm Locally Converges

The studies up to this step show that blind equalization algorithms using Kalman filtering seem to work better than other algorithms in terms of convergence rate for linear, time invariant, stable channel models and binary sources using BPSK. However they still suffer from the local convergence problem as seen for the CMA algorithms.

III. BLIND EQUALIZATION ALGORITHM BY USING PARALLEL KALMAN FILTERING

As seen in the previous chapter, the algorithm based on the extended Kalman filter for blind equalization, suffers from the local convergence problem, as most of the other CMA algorithms. For the algorithm to be applicable, we need to address this problem. As a candidate solution we choose a particular parallel Kalman filtering approach, which is described in detail in [Ref.4]. In this chapter, we will present this algorithm and discuss the results of our studies with this algorithm.

A. DEVELOPMENT OF THE PARALLEL KALMAN FILTERING ALGORITHM

This algorithm is an approximation to the optimum maximum a posteriori (MAP) sequence estimator for a priori unknown channels. The sequence probabilities can be computed using the innovations derived from a bank of Kalman filters.

In the discrete-time channel and signal model, $x(t)$ denotes the output of a matched filter at time t, $s(t)$ is the current transmitted symbol and $\{a_n(t)\}$ represent the effective time varying channel coefficients. For BPSK, the $s(t)$ are real valued, taking on values $\{+1,-1\}$. The channel coefficients $a_n(t)$ represent the convolution of the actual intersymbol interference (ISI) channel impulse response with that of a prewhitening filter, which is included to insure that the additive noise samples, $n(t)$, are uncorrelated.

$$x(t) = \sum_{n=0}^{N_b} a_n(t)\ s(t-n) + n(t) \qquad (3.1)$$

The additive noise sequence is complex white Gaussian with variance σ_n^2 and N_b+1 is the length of the channel impulse response.

For convenience in the derivation, the following sequences are also defined for the ith of M^{t+1} possible sequences, where M is the symbol alphabet size:

The *cumulative measurement sequence*,

$$X_t = \{x(t), x(t-1), \ldots, x(0)\}$$

A *cumulative data sequence*,

$$D_{i,t} = \{d_i(t), d_i(t-1), \ldots, d_i(0)\}$$

A *data subsequence*, comprising the data symbols associated with the channel coefficient vector $\underline{a}(t)$ at time t,

$$D_{i,t,N_b} = \{d_i(t), d_i(t-1), \ldots, d_i(t-N_b)\}.$$

In the development of the Kalman filter channel estimator, it is assumed that the coefficient $b_n(t)$ evolve according to the following complex Gaussian autoregressive(AR) process model

$$\underline{b}(t+1) = \underline{F}\,\underline{b}(t) + \underline{w}(t)\ , \qquad (3.2)$$

with the coefficient vector defined by

$$\underline{b}(t) = [\, b_1(t), b_2(t-1), \ldots, b_{N_s}(0) \,]^T \,.$$

In Equation(3.2) $\underline{F}$ is the one step transition matrix and $\underline{w}(t)$ is a white Gaussian process with covariance matrix $\underline{Q}$.

The optimum MAP sequence estimator can be written in the following recursive form for the assumed channel and signal models

$$p(D_{i,t} \mid X_t) = \frac{1}{c}\; p(x(t) \mid D_{i,t}, X_{t-1})\;\; p(D_{i,t-1} \mid X_{t-1}) \qquad (3.3)$$

for $i = 1,2,\ldots,M^{t+1}$, where $p(D_{i,t} \mid X_t)$ represents the probability of the ith possible data sequence given cumulative measurements X_t, and c is a normalization constant. The likelihood $p(x(t) \mid D_{i,t}, X_t)$ is given in terms of the Kalman filter innovations as

$$p(x(t) \mid D_{i,t}, X_{t-1}) = N(\hat{s}_i(t), \sigma_i^2(t \mid t-1)) \;, \qquad (3.4)$$

where $N(x, \sigma_x^2)$ denotes a Gaussian density with mean x and covariance σ_x^2. The estimated signal $\hat{s}_i(t)$, is given in terms of the conditional channel estimates according to

$$\hat{s}_i(t) = \sum_{n=0}^{N_s} \hat{b}_{i,n}(t \mid t-1)\;\; d_i(t-n) \;, \qquad (3.5)$$

where $\hat{b}_{i,n}(t \mid t-1)$ is generated by the Kalman filter equations discussed in Chapter II. The estimate $\hat{b}_{i,n}(t \mid t-1)$ can be shown to be exactly equal to the conditional mean of the channel coefficients $a_n(t)$, under the AR process model in Equation(3.2) when conditioned on data sequence $D_{i,t}$. Thus,

$$\hat{b}_{i,n}(t|t-1) = E[b_n(t) \mid D_{i,t}, X_{t-1}] \quad . \tag{3.6}$$

From Equations (3.3) and (3.4), it is seen that the optimum MAP sequence estimator requires a bank of M^{L+1} Kalman filter channel estimators, each conditioned on a different $D_{i,t}$. The MAP probabilities of each sequence are then obtained as a product of the corresponding innovations likelihoods.

By using the concept of reduced state sequence estimation (RSEE), the algorithm becomes as follows (more detailed derivation can be found in [Ref.4]):

i. Define observation vectors

$$h_i(t) = [\, d_i(t), d_i(t-1), \ldots, d_i(t-N_b)\,] \tag{3.7}$$

ii. Compute conditional innovations covariances

$$\sigma_i^2(t|t-1) = h_i(t)\; h_i^T(t) + \sigma_n^2 \tag{3.8}$$

iii. Compute signal estimates

$$\hat{s}_i(t) = \sum_{n=0}^{N_b} \hat{b}_{i,n}(t|t-1)\; d_i(t-n) \tag{3.9}$$

iv. Compute conditional measurement update

$$\hat{b}_i(t|t) = \hat{b}_i(t|t-1) + \frac{\alpha}{\sigma_i^2(t|t-1)}\; h_i^T(t)\;\; (x(t) - \hat{s}_i(t)) \tag{3.10}$$

v. **Update weighing probabilities**

$$p(D_{i,j,N_b} \mid X_t) = \frac{1}{C} N(\hat{s}_i(t), \sigma_i^2(t \mid t-1)) \sum_{j:D_{j,j-1,N_b} \in D_{i,j,N_b}} p(D_{j,j-1,N_b} \mid X_{t-1}) \quad (3.11)$$

vi. **Compute one step predictions**

$$\hat{b}_i(t+1 \mid t) = \sum_{j:D_{j,j,N_b} \in D_{i,j-1,N_b}} \hat{b}_j(t \mid t) \frac{p(D_{j,j,N_b} \mid X_t)}{\sum_{m:D_{m,j,N_b} \in D_{i,j-1,N_b}} p(D_{m,j,N_b} \mid X_t)} \quad (3.12)$$

B. ADMISSIBILITY OF THE BLIND EQUALIZATION BY PARALLEL KALMAN FILTERING

In order to be consistent with the comparison of the algorithms, we will consider only linear, time invariant, stable, channel models and BPSK. Then, $s(t) \in \{+1,-1\}$ therefore M=2 ,and the length of the channel impulse response is $N_b+1=3$ (assumed). Now $M^{Nb+1}=2^3=8$ different observations are needed. These are:

$$h_0(t) = \{-1, -1, -1\}$$
$$h_1(t) = \{-1, -1, +1\}$$
$$h_2(t) = \{-1, +1, -1\}$$
$$h_3(t) = \{-1, +1, +1\}$$
$$h_4(t) = \{+1, -1, -1\}$$
$$h_5(t) = \{+1, -1, +1\}$$
$$h_6(t) = \{+1, +1, -1\}$$
$$h_7(t) = \{+1, +1, +1\}$$

Assuming $\sigma_n^2=0.01$ (small) and $\alpha=1$ (according to [Ref.4] $0<\alpha<2$), also

$$N(\hat{s}_i(t), \sigma_i^2(t|t-1)) = \frac{1}{\sqrt{2\pi\sigma_i^2(t|t-1)}} \exp - \frac{(x(t)-\hat{s}_i(t))^2}{2\sigma_i^2(t|t-1)} \quad (3.13)$$

Substitution of all these parameters into the algorithm discussed in Section A yields the performance with various fixed equalizer parametrization, which is shown in Figure (3.1).

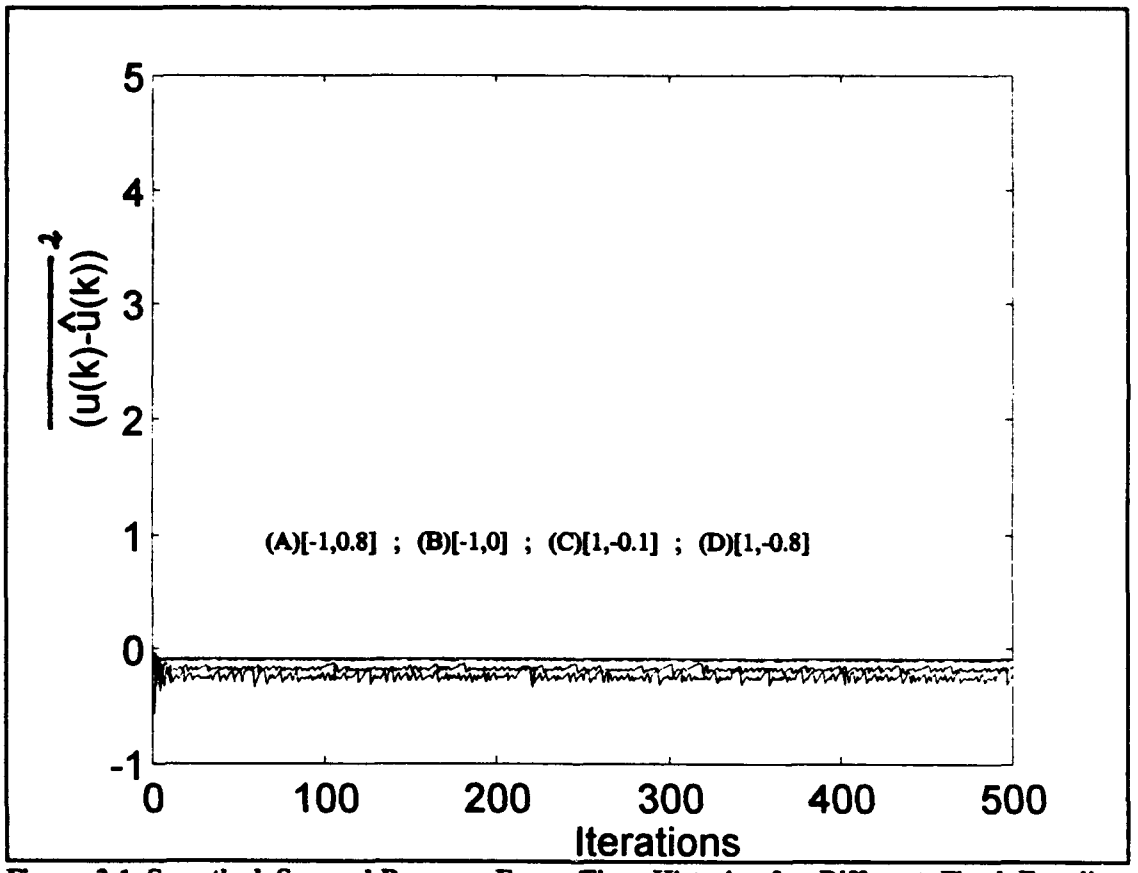

Figure 3.1 Smoothed Squared-Recovery-Error Time Histories for Different Fixed Equalizer Parametrizations [(b_0 ,b_1)]

In Figure (3.1) all (b_0,b_1) settings result in perfect performance though $H(z) \neq 1$. Also, the phase ambiguity problem which occurs in the extended Kalman filtering

algorithm is solved. The exponential rise does not appear since we did not use a smoothing filter, and (b_0,b_1) setting (0,1) does not appear in the figure since the model is undetermined for this case.

The performance of the algorithm for different initializations and the (b_0,b_1) setting of (1,-0.6) is also studied. The initial values are shown as (x) and the final values are shown as (o) in Figure (3.2). There is no convergence to local minima for the model, so optimum results are achieved.

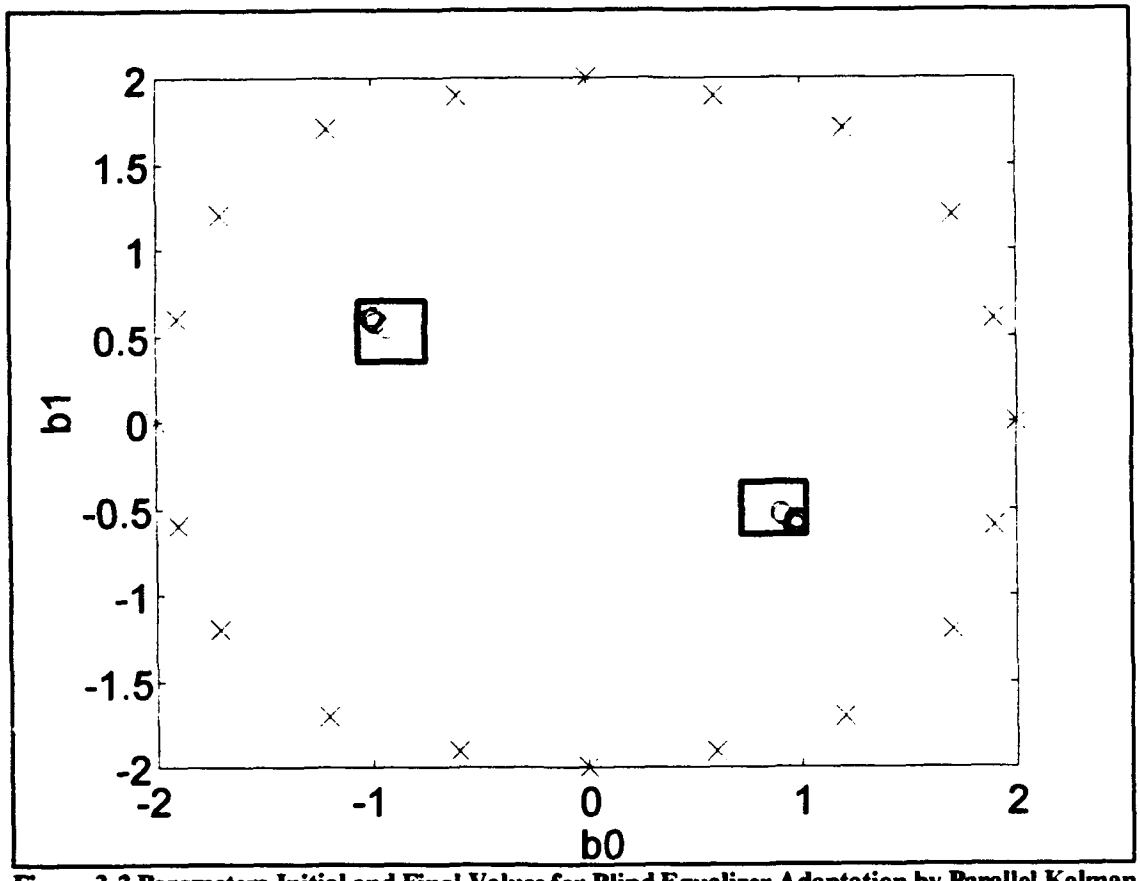

Figure 3.2 Parameters Initial and Final Values for Blind Equalizer Adaptation by Parallel Kalman Filtering

However, during our studies we observed that this algorithm does not work properly all the time. It fails to converge to global minima 60% of the time. Also, it

can not work as well as the extended Kalman filtering approach in nonminimum phase systems. When the algorithm converges to local minima, there is no way to determine the situation with this algorithm as in the Kalman filtering approach. So the algorithm becomes unreliable, since we can not decide whether the global minima have been reached or not by any means. The usage of multiple Kalman filters is another disadvantage of this algorithm, when the cost is considered.

IV. APPLICATION OF HIDDEN MARKOV MODELS TO BLIND EQUALIZATION ALGORITHMS WITH KALMAN FILTERING

In communication systems the transmitted sequence $s(t)$ is not a random independent identically distributed (i.i.d.) sequence. Indeed forward error correction (FEC) coding is applied to a message sequence $u(t)$, in order to produce the transmitted sequence $s(t)$. At this point we realized that, none of the algorithms known so far uses the statistics of the transmitted sequence $s(t)$, which might be useful for the channel equalization. Obviously, if we decide to use the statistics of the transmitted sequence $s(t)$ in equalization, we can combine the equalizer with the FEC decoder. It is a fact that, most of the FEC coding systems which are used in communication systems, are convolutional coders and decoders. They mostly use the Viterbi soft decoding technique, which is an optimum estimation for Markov models. So in our new approach we will try to combine the Kalman filtering algorithm with hidden Markov models.

A. HIDDEN MARKOV MODELS

Consider a system that may be described at any time as being in one of a set of N distinct states indexed by {1,2,3,...,N}. At regularly spaced, discrete times, the system undergoes a change of state according to a set of probabilities associated with the state. We denote the time instants associated with state changes as t=1,2,3,... , and we denote

the actual state at time t as q_t. For a discrete time, first order, Markov model, the probabilistic dependence is truncated to just the preceding state.

$$p[q_t=j \mid q_{t-1}=i,\ q_{t-2}=k,\ q_{t-3}=l,\ \ldots]=p[q_t=j \mid q_{t-1}=i] \qquad (4.1)$$

Furthermore, those processes in which the right hand side of Equation (4.1) is independent of time, lead to the set of state transition probabilities a_{ij} of the form

$$a_{ij}=p[q_t=j \mid q_{t-1}=i] \ , \quad 1 \le i,j \le N \qquad (4.2)$$

with the following properties

$$\begin{aligned} a_{ij} &\ge 0 \quad \forall\ j,i \\ \sum_{j=1}^{N} a_{ij} &= 1 \ . \quad \forall\ i \end{aligned} \qquad (4.3)$$

The notation:

$$\pi_i=p[q_1=i] \quad 1 \le i \le N \qquad (4.4)$$

is used to denote the initial state probabilities.

The concept of Markov models can be extended to include the case in which the observation is a probabilistic function of the state, so the resulting model (which is called a hidden Markov model) is a doubly embedded stochastic process with an underlying stochastic process that is not directly observable (it is hidden) but can be observed only through another set of stochastic processes that produce the sequence of observations. As a result a hidden Markov model (HMM) for discrete symbol observations can be characterized by the following elements:

i). N, the number of states in the model,

ii). M, the number of distinct observation symbols per state. We denote the individual symbols as

$$V = \{v_1, v_2, v_3, \ldots, v_M\}$$

iii). The state transition probability matrix $A = \{a_{ij}\}$ where

$$a_{ij} = p[q_{t+1} = j \mid q_t = i] \qquad 1 \le i, j \le N \tag{4.5}$$

iv). The observation symbol probability distribution, $B = \{b_j(k)\}$, in which

$$b_j(k) = p[o_t = v_k \mid q_t = j] \qquad 1 \le k \le M \tag{4.6}$$

defines the symbol distribution in state j, where j=1,2,3,...,N. Also a typical observation sequence of the model can be shown as

$$O = \{o_1, o_2, o_3, \ldots, o_T\}$$

v). The initial state distribution $\pi = \{\pi_i\}$ in which

$$\pi_i = p[q_1 = i] \qquad 1 \le i \le N \tag{4.7}$$

So a complete specification of a HMM requires specification of two model parameters, N and M, specification of observation symbols, O, and the specification of the three sets of probability measures A, B and π. The compact notation

$$\lambda = (A, B, \pi) \tag{4.8}$$

is used to indicate the complete parameter set of model. Given the appropriate values of N, M, A, B and π, the HMM can be used as a generator to give an observation sequence

$$O=(o_1, o_2, o_3, \ldots, o_T) \qquad (4.9)$$

where each observation o_t is one of the symbols from V, and T is the number of observations in the sequence. This model is the main model used for FEC coding, and it works as follows:

1. Choose an initial state $q_1=i$ according to the initial state distribution π,
2. Set $t=1$,
3. Choose $o_t=v_k$ according to the symbol probability distribution in state i, i.e., $b_j(k)$,
4. Transit to a new state $q_{t+1}=j$ according to the state transition probability distribution for state i, i.e., a_{ij},
5. Set $t=t+1$; return to step 3 if $t<T$; otherwise, terminate the procedure.

Now, let us approach the problem from our point of view. We have the observation sequence O, which is produced by the model, described above, and we know the model λ. We need to choose a corresponding state sequence $q=(q_1, q_2, \ldots, q_t)$ that is optimal in some sense (i.e., best explains the observations).

We can define the posteriori probability as

$$\gamma_t(i)=p[q_t=i \mid O,\lambda]=\frac{p[O, q_t=i \mid \lambda]}{p[O \mid \lambda]}=\frac{p[O, q_t=i \mid \lambda]}{\sum_{i=1}^{N} p[O, q_t=i \mid \lambda]} \qquad (4.10)$$

Using $\gamma_t(i)$ we can solve for the individually most likely state q_t^* at time t, as

$$q_t^* = arg\ \min_{1 \le i \le N} [\gamma_t(i)] \qquad 1 \le t \le T \tag{4.11}$$

Using this criteria iteratively the single best state sequence (path) can be found. To find the single best state sequence q, for the given observation sequence O, we need to define the quantity

$$\delta_t(i) = \max_{q_1, \ldots, q_{t-1}} p[q_1, q_2, \ldots, q_{t-1}, q_t = i, o_1, o_2, \ldots, o_t | \lambda] \tag{4.12}$$

that is the best score (highest probability) along the single path, at time t, which accounts for the first t observations and ends in state i. By induction we can compute it recursively as

$$\delta_t(i) = \max_j \ \delta_{t-1}(j)\ p[q_t = i | q_{t-1} = j]\ p[o_t | q_t = i] \tag{4.13}$$

To actually retrieve the state sequence, we need to keep track of the argument that maximized Equation (4.13), for each t and j. The formal technique for doing the job is based on dynamic programming methods, and is called the Viterbi algorithm. [Ref.5]

B. COMBINING KALMAN FILTERING ALGORITHMS WITH HMM

In general, a message sequence $u(t)$ is coded by a HMM. The block diagram of an overall communication scheme is shown in Figure (4.1). The discrete coder states $z(t)$ where $z(t) \epsilon Z = \{1, 2, 3, \ldots, L\}$ and the output sequence $s(t)$ are dependent on the older coder states and the input sequence $u(t)$ to the coder. Thus,

$$\begin{aligned} z(t+1) &= F[z(t), u(t)] \\ s(t) &= G[z(t), u(t)] \end{aligned} \tag{4.14}$$

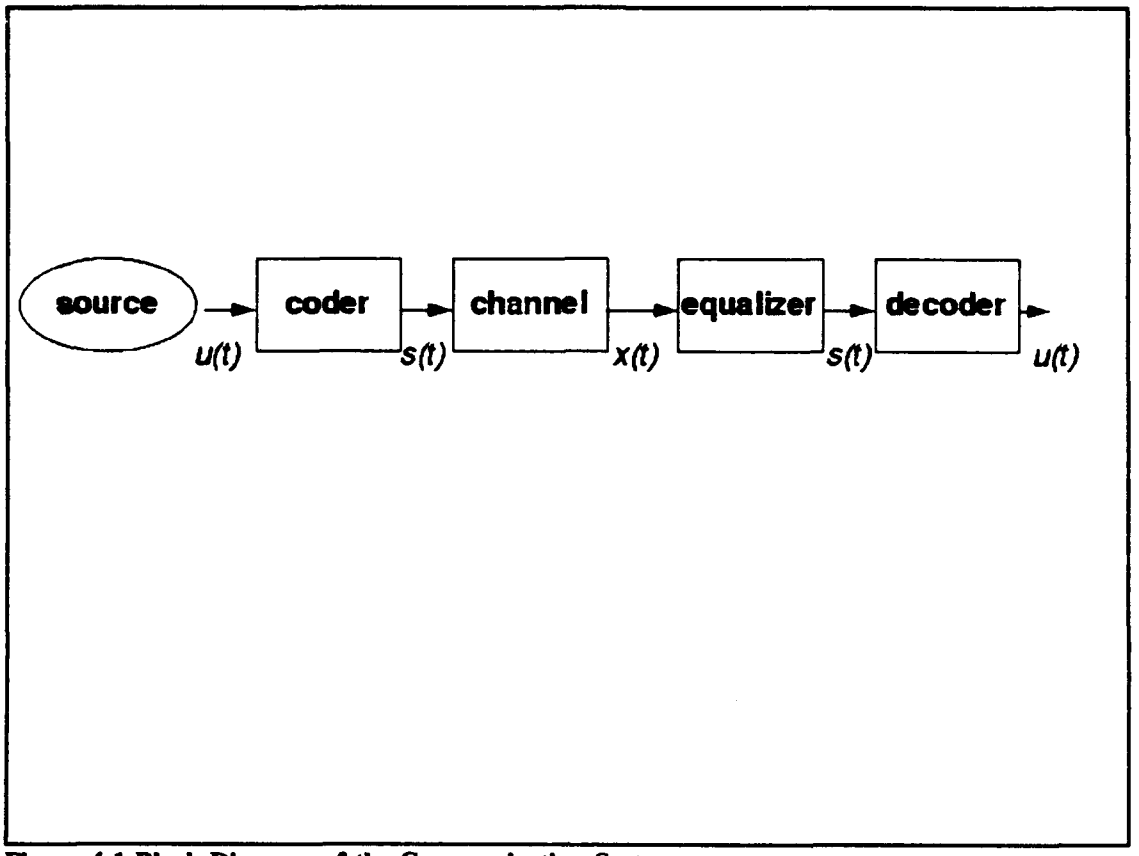

Figure 4.1 Block Diagram of the Communication System

As we stated in Chapter II before; the vector X_t represents the received signal values and the vector W_t represents the state coefficients of the equalizer. So we can show the output of the equalizer as

$$y(t) = \hat{s}(t) = W_t^T(t)\ X_t(t) + e(t) \qquad (4.15)$$

where $e(t)$ is estimation error described in Chapter I.

In extended Kalman filtering approach we assumed the channel changes slowly and the model is defined in Equation (2.21). We account for drifts in channel parameters by the recursion

$$W_t(t+1) = W_t(t) + v(t) \qquad (4.16)$$

Then combining the Equations (4.14), (4.15), and (4.16) we can represent the overall model of equalizer and decoder as

$$\begin{aligned} z(t+1) &= F[z(t), u(t)] \\ W_t(t+1) &= W_t(t) + v(t) \\ 0 &= G[z(t), u(t)] - W_t^T(t)\ X_t(t) + e(t) \end{aligned} \qquad (4.17)$$

The difficulty with this state space model is that the state $[\underline{Z}(t), \underline{W}(t)]$ is a mixture of a discrete component $\underline{Z}(t)$ and a continuous one $\underline{W}(t)$.

The estimation algorithm is based on the fact that, given the sequences

$$Z_t(t) = [z(1), z(2), \ldots, z(t)]$$

$$Y_t(t) = [y(1), y(2), \ldots, y(t)]$$

the estimate of w

$$\hat{W}_t = E[\ W_t \mid Z_t\ ,\ Y_T\ ] \qquad (4.18)$$

is well defined, and recursively computable by standard Kalman filtering techniques.

An overall recursion along the same lines as the estimation for hidden Markov models can be devised for this case.

The suboptimal optimization is based on the definition

$$\delta_t(i) = \max_j\ p(\hat{Z}_t^j(t-1), z(t) = i, Y_t(t)) \qquad (4.19)$$

where $\hat{z}_t^j(t-1)$ is the optimum path up to state $z(t-1) = j$. By induction we can update $\delta_t(i)$ as

$$\delta_t(i) = \max_j \left[p(y(t) \mid \hat{Z}^j_{t-1}(t-1), z(t)=i, Y_t(t-1)) p(z(t)=i \mid z(t-1)=j)\, \delta_{t-1}(j) \right] \quad (4.20)$$

and keep track of the indices on the optimal path

$$\hat{j} = \arg\max \left[p(y(t) \mid \hat{Z}^j_{t-1}(t-1), z(t)=i, Y_t(t-1)) p(z(t)=i \mid z(t-1)=j)\, \delta_{t-1}(j) \right] \quad (4.21)$$

The optimal path up to state $z(t)=i$ is then updated as

$$\hat{Z}^j_t(t) = \hat{Z}^j_{t-1}(t-1) \bigcup z(t)=i \quad (4.22)$$

Also we know all the possible previous states j and the state coefficient estimates of the channel up to this state $\hat{w}_{t-1}(j)$ is already computed by Kalman filtering as stated in Equation (4.18). So we can determine the optimum estimates of the channel state coefficients by the following procedure:

$$\hat{W}_t(i \mid j) = \hat{W}_{t-1}(j) + \kappa_{t-1}[G[j, u(t)] - \hat{W}^T_{t-1}(j)\, X_t(t) + e(t) \quad (4.23)$$
$$\hat{W}_t(i) = \text{most likely}_j\ \hat{W}_t(i \mid j)$$

C. FEASIBILITY OF HMM AND KALMAN FILTERING APPROACH TO BLIND EQUALIZATION

This new algorithm works almost perfect in linear, time invariant, stable, channel models and BPSK. This new algorithm seems to be more capable than the previous ones, and we applied it to more complex channels. We have used different feasible FEC encoders (with different rates and constraint lengths), BPSK signaling, additive white

Gaussian noise (AWGN) and different signal to noise ratios (SNR) in the studies. The results are presented in Figures (4.2) through (4.4).

Since the perfect equalization requires

$$a(t) * b(t) = 1 \quad ,$$

the complete impulse response of the combination of the channel and the equalizer must appear as an impulse function, which is the case in our studies (shown at the left upper corner of the figures).

By looking at the results, this new algorithm recovers the sent message without errors after a couple of iterations, which are very small compared to the length of the transmitted sequence. This algorithm works much better in high SNR values. Errors do occur if SNR drops under 5 dB. but further studies are needed to determine the exact margin. With this kind of performance, this algorithm can also be a candidate to address the channel drifting problem.

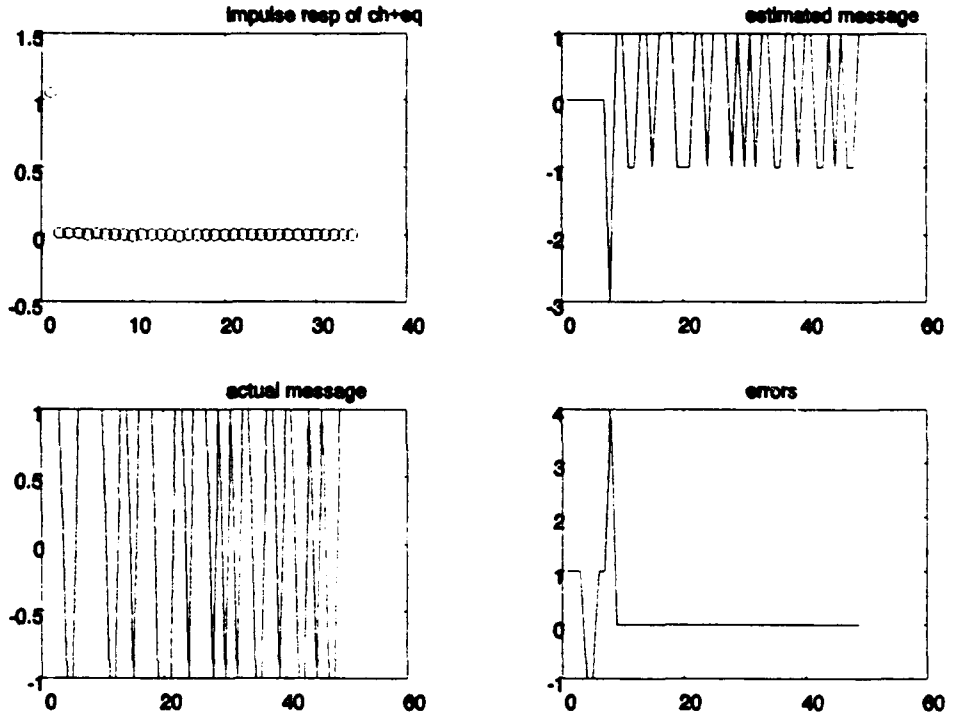

Figure 4.2 Channel and Message Estimation for Rate=1/2,Con.Len=3 Encoder with High SNR

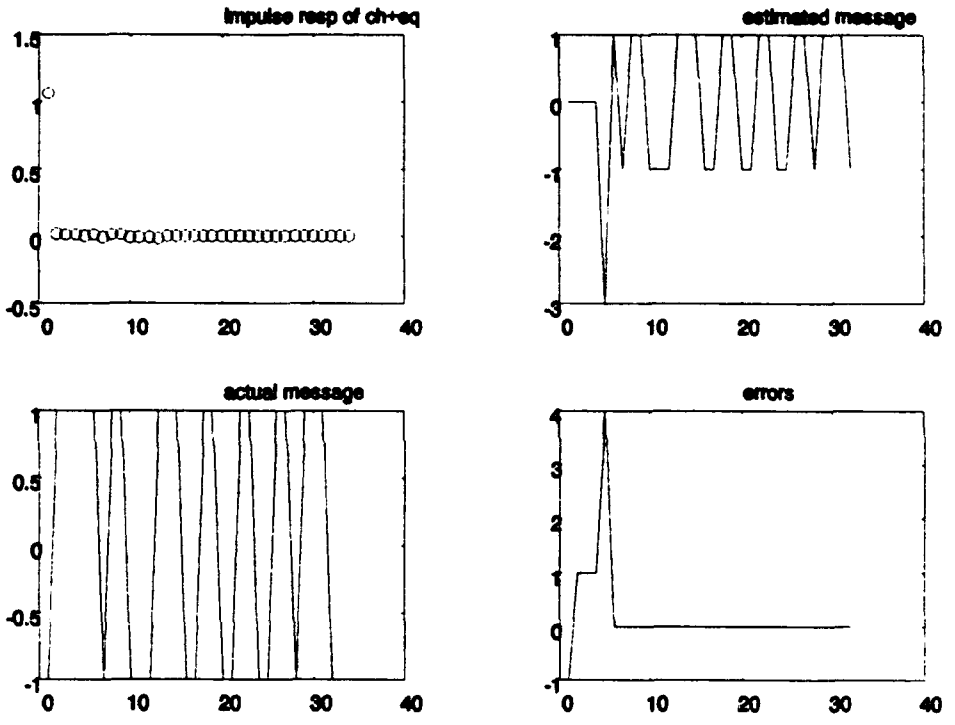

Figure 4.3 Channel and Message Estimation for Rate=1/3, Con.Len=3 Encoder with High SNR

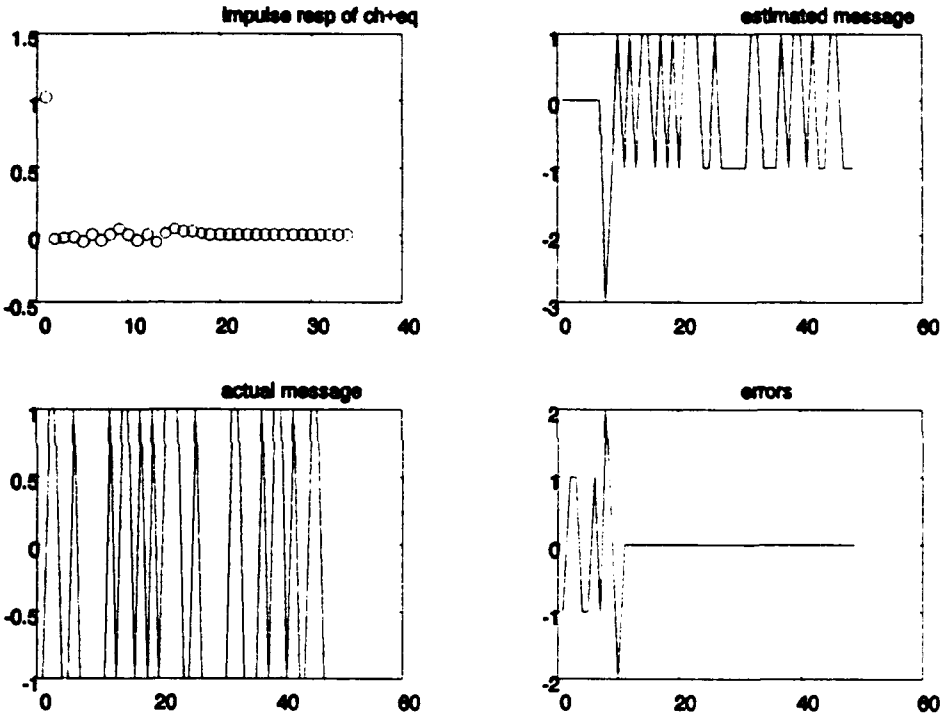

Figure 4.4 Channel and Message Estimation for Rate=1/2,Con.Len=7 Encoder with High SNR

V. SIMULATION AND RESULTS

We have simulated a flat Rayleigh fading channel and tried our new algorithm on it, in order to see its feasibility in overall communication systems. The block diagram of our simulation scheme is shown in Figure (5.1).

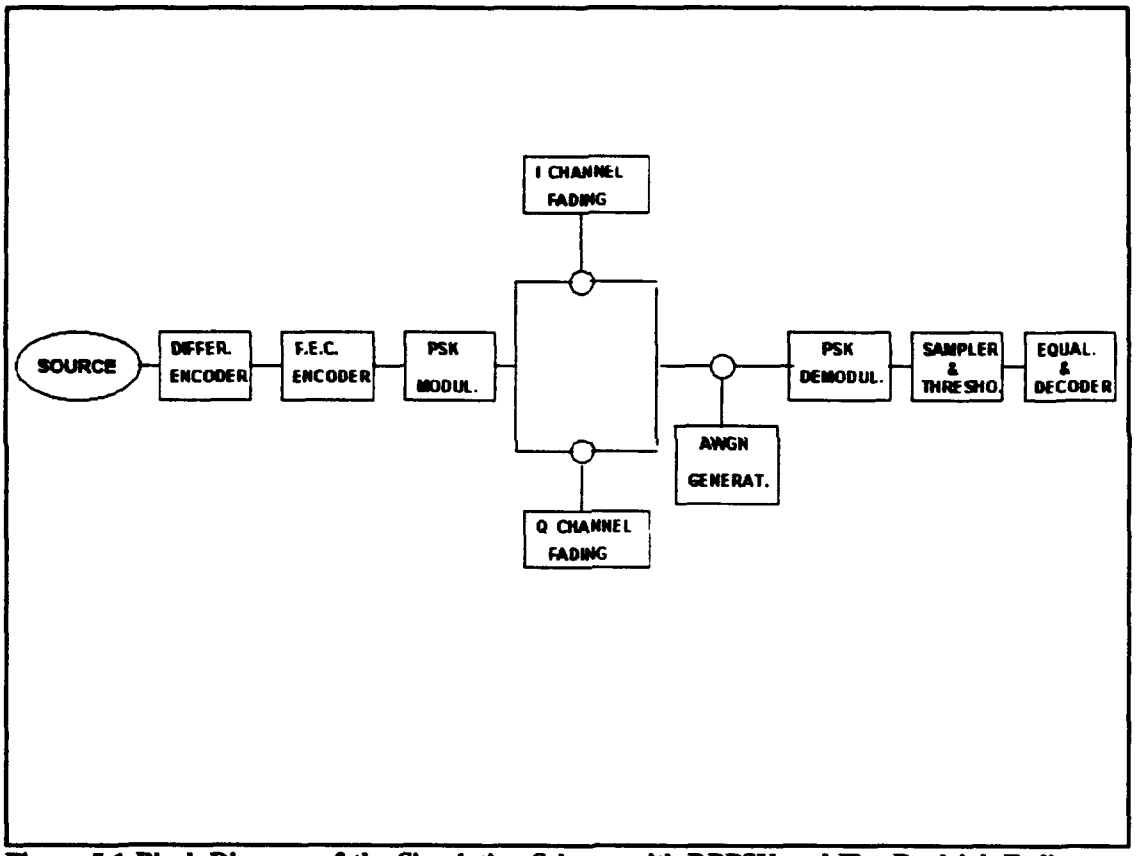

Figure 5.1 Block Diagram of the Simulation Scheme with DBPSK and Flat Rayleigh Fading

In the simulation DBPSK is used as a modulation scheme. DBPSK has hardware simplicity and does not require phase coherency. It is popular for channels where the phase shift changes slowly compared to the bit duration.

Also the industrial standard FEC encoder, with rate 1/2, constraint length 7 and $(133,171)_8$ connections is employed. The encoder and its connections are illustrated in Figure (5.2).

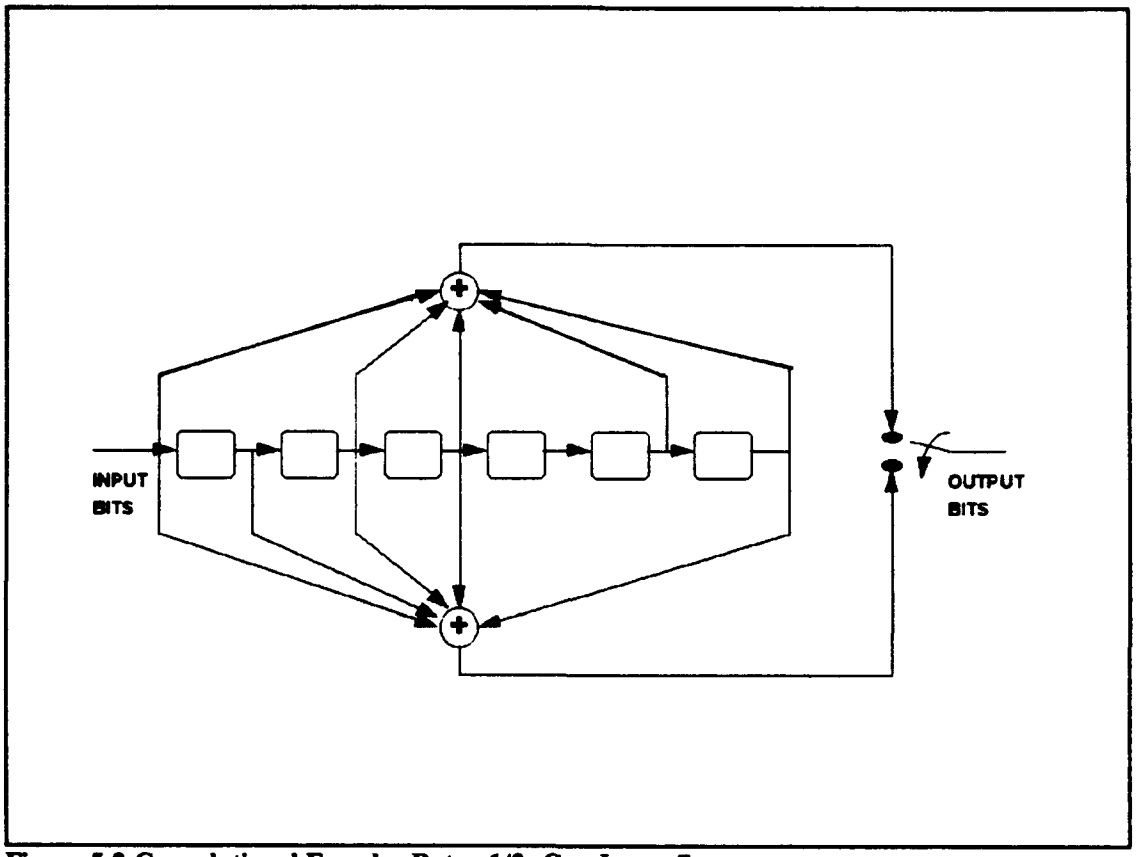

Figure 5.2 Convolutional Encoder Rate=1/2, Con.Len.=7

The flat Rayleigh fading channel model is also used and illustrated in Figure (5.3). This model corresponds to a single path Rayleigh fading channel. In this channel there is no frequency selective distortion, and the channel is very suitable to represent mobile communication schemes. More detailed explanation about this model and its computational representation can be found in [Ref.6].

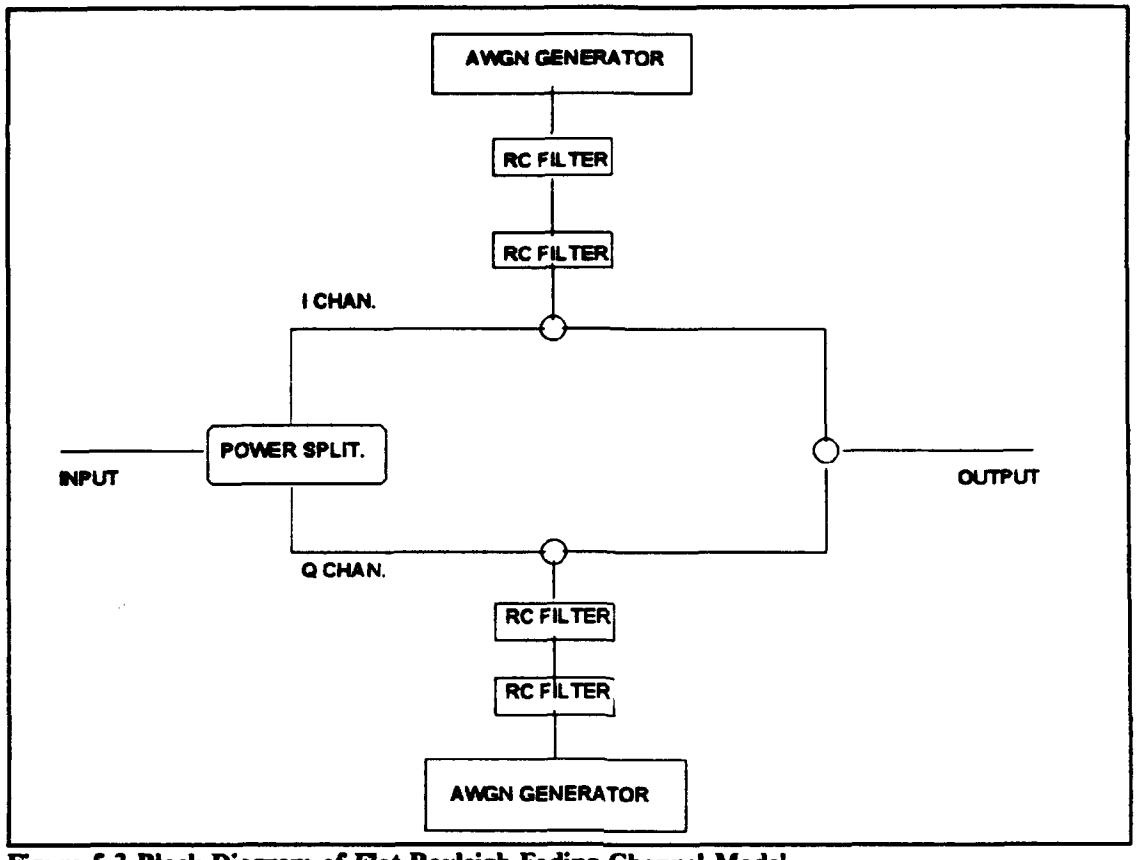

Figure 5.3 Block Diagram of *Flat Rayleigh Fading* Channel Model

The results of the simulation using a signal to noise ratio (SNR) of 13 dB. is given in Figure (5.4). As it can be seen from the results of our simulation, the combination of equalizer and decoder by HMM and Kalman filtering approach performs fairly good in complex channels and under channel fading conditions.

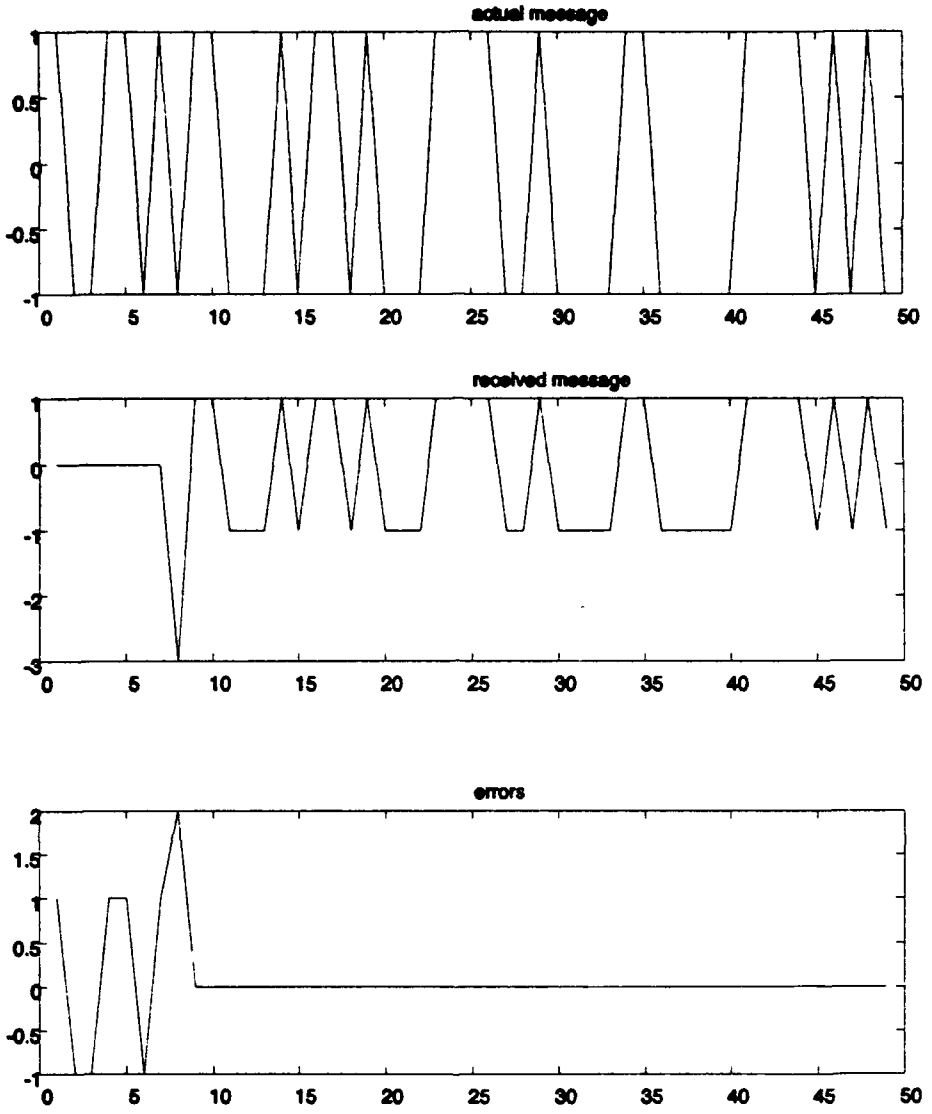

Figure 5.4 Estimation of the Message Sequence in Flat Rayleigh Fading Channel Using DBPSK and Rate=1/2 FEC Code

VI. CONCLUSIONS

An alternative algorithm for blind equalization of digital communication channels, is derived and studied by using extended Kalman filtering and hidden Markov models. This new algorithm seems more efficient since it takes the statistics of the transmitted sequence into consideration. It seems to be more robust with respect to local minima, and channel drifting problems.

Hardware implementation of this algorithm might be cost effective since it combines two major components of a receiver, namely the equalizer and the decoder into a single component.

Future studies can determine the overall performance of this algorithm under different channel conditions and for different communication schemes.

APPENDIX A. MATLAB SOURCE CODES

The algorithms described in Chapters II,III and IV were implemented using MATLAB software and are listed below.

A. BLIND EQUALIZER FOR LINEAR, TIME-INVARIANT, STABLE CHANNELS BY USING EXTENDED KALMAN FILTER

```
% Filename : eqkalm.m
% Title    : Blind equalizer for linear,time-invariant,stable channels  by using Kalman filtering
%            algorithm
% Date of last revision : 15 Jul 1993
% Comments :  This program produces an array of  random digital  signal values,
%             (either + or -1) to represent the message signal.The length of message signal
%             depends on the sampling amount m(of course bigger m makes better
%             estimation). Then  this signal is applied to a channel such as:[x(t) x(t-1) ...
%             x(t-n)] where the coefficiants defined by user.To estimate  the equalizer
%             coefficiants, a rectangular window over the received signal values, is going to
%             be used such as[x(t) x(t-1) ... x(t-n)] where (n+1) represents the # of values
%             taken into account by the window (and by the equalizer).In this way program
%             tries to predict real coefficients of channel.If values match equalizer design
%             will be perfect.
%             Averaged squared recovery error due to the channel and qualizer,performance
%             due to the different initializations and normalized squared error distributions
%             can be plotted.
% Input variables :
%             f : The coefficients of the channel as a vector
%             m : Sampling amount
%             wt: Initial values of the coefficients at the equalizer as a column vector
%             sel: Selection to continue with the calculation
%             op: Options to plot different schemes
% Output variables :
%             b : The estimated values of the channel coefficients by equalizer
% Associated functions : None

clear;
f=input('NOW ENTER the coefficients of your channel as vector.:');
m=input('   ENTER the value of m (sampling amount).....:');
sel=input('   ENTER 1 to continue.....:');
```

```
%%% Produce message %%%
n=length(f)-1;
y=sign(randn(1,(m+n)));

while sel==1
 wt=.01*ones((n+1),(m+n));
 wt(:,n)=input('ENTER the initial coefficiants for your equalizer as column vector..:');
%%% Arrangements %%%
 xt=zeros((n+1),1);
 et=zeros(1,m);
 errt=zeros(1,m);
 ay=eye((n+1),(n+1));
 pt=diag([1e-1,1e-1]);
 x=zeros(1,(n+1));
%%% Pass message through the channel %%%
 for i=n+1:m+n
  x(i)=(-f(2:n+1).*x(i-1:-1:i-n)+y(i))/f(1);
 end
%%% Kalman filtering %%%
 for j=n+1:m+n
  xt=(x(j:-1:j-n))';
  wt1=ay*wt(:,j-1);
  yt1=wt(:,j-1)'*xt*xt'*wt(:,j-1);
  c=2*wt(:,j-1)'*xt*xt';
  pt1=ay*pt*ay';
  gt=pt1*c'*inv(c*pt1*c'+.01);
  pt=(ay-(gt*c))*pt1;
  wt(:,j)=wt1+(gt*(1-yt1));
  et(:,j-n)=1-yt1;
  errt(:,j-n)=et(:,j-n)/sqrt(c*pt*c'+.01);
 end
%%% Results %%%
 b=wt(:,m+n)'
 pause
%%% Estimation errors %%%
 for k=n+1:n+m
  r(k)=(b(1:n+1)*x(k:-1:k-n)');
  s(k)=(y(k)-r(k))^2;
 end
%%% Plots %%%
 disp('For the graphs you have the following options:');
 disp('ENTER 1 to see aver. squ. error due to the chan. & equa.');
 disp('ENTER 2 to see performance due to the different initializations');
 disp('ENTER 3 to see normalized squared error distributions');
 op=input('ENTER 0 to quit.....:');
 while op==1,
  plot(s(n+1:n+m))
```

```
  grid;
  xlabel('Iterations');
  ylabel('(u(k)-u(k)) ');
  pause
  disp('For the graphs you have the following options:');
  disp('ENTER 2 to see performance due to the different initializations');
  disp('ENTER 3 to see normalized squared error distributions');
  op=input('ENTER 0 to quit.....:');
 end
 while op==2,
   axis([-2 2 -2 2]);
   plot(wt(1,:),wt(2,:))
   grid;
   xlabel('b0');
   ylabel('b1');
   pause
   disp('For the graphs you have the following options:');
   disp('ENTER 1 to see aver. squ. error due to the chan. & equa.');
   disp('ENTER 3 to see normalized squared error distributions');
   op=input('ENTER 0 to quit.....:');
 end
 while op==3,
   [n2,x2]=hist(errt,30);
   k2=-4:.01:4;
   gt=1/sqrt(2*pi)*exp((-(k2.^2))/2);
   m2=max(gt);
   [xb2,yb2]=bar(x2,((n2/max(n2))*m2));
   plot(k2,gt,xb2,yb2);
   ort=mean(errt);
   var=(std(errt)^2);
   gtext(['mean=',num2str(ort)]);
   gtext(['var=',num2str(var)]);
   pause
   disp('For the graphs you have the following options:');
   disp('ENTER 1 to see aver. squ. error due to the chan. & equa.');
   disp('ENTER 2 to see performance due to the different initializations');
   op=input('ENTER 0 to quit.....:');
 end

 disp('   ENTER 0 to exit to matlab.....:');
 sel=input('   ENTER 1 to continue.....:');

end

end;
```

B. BLIND EQUALIZER FOR LINEAR, TIME-INVARIANT, STABLE CHANNELS BY USING PARALLEL KALMAN FILTERS

```
% Filename : parkal.m
% Title    : Blind equalizer for linear,time-invariant,stable channels
%            by using parallel Kalman filtering algorithm
% Date of last revision : 18 Jul 1993
% Comments : This program produces an array of  random digital
%            signal values, (either + or -1) to represent the message
%            signal.The length of message signal depends on the sampling
%            amount m(of course bigger m makes better estimation).Then
%            this signal is applied to a channel such as:[x(t) x(t-1) ...
%            x(t-n)] where the coefficiants defined by user.To estimate
%            the equalizer coefficiants, a rectangular window over the
%            received signal values, is going to be used such as[x(t)
%            x(t-1) ... x(t-n)] where (n+1) represents the # of values
%            taken into account by the window (and by the equalizer).
%           Also each symbol in the message alphabet is assigned to
%           a specific Kalman filter.In this way program tries to
%           predict real coefficients of the channel.At the end correct
%           channel parameter estimations appear at least at one of the
%           Kalman filters in the bank.If values match equalizer design
%           will be perfect.( Program works for alphabet size 4 only)
%            Averaged squared recovery error due to the channel and
%            equalizer,performance due to the different initializations
%            and normalized squared error distributions can be plotted.
% Input variables :
%            f : The coefficients of the channel as a vector
%            m : Sampling amount
%            bi: Initial values of the coefficients at the equalizer as
%                a column vector
%            sel: Selection to continue with the calculation
%            sec: Options to plot different schemes for different filters
% Output variables :
%            b* : The estimated values of the channel coefficients by
%                 associated Kalman filter *.
% Associated functions : None

clc;

clear;
f=input('NOW ENTER the coefficients for the channel as vector(2 elements).:');
m=input('    ENTER the value of m (sampling amount).....:');
sel=input('    ENTER 1 to continue.....:');
hold off
```

```
while sel==1
 bi=input('ENTER the initial coefficiants for your equalizer as column vector..:');
 n=length(f)-1;
 y=sign(randn(1,(m+n)));
 r=zeros(1,(n+1));
 z0=ones(1,(n+1));
 z1=ones(1,(n+1));
 z2=ones(1,(n+1));
 z3=ones(1,(n+1));
 et=zeros(1,m);
 errt=zeros(1,m);
 for i=n+1:m+n
   r(i)=(f(1:n+1)*y(i:-1:i-n)');
%   r(i)=(-f(2:n+1).*r(i-1:-1:i-n)'+y(i))/f(1);      % Nonrecursive case
 end

 b0=bi;                          %
 h0=[-1 -1];                     %
 sig0=h0*h0'+.01;                %FOR EACH FILTER
 p0=0.25;                        %

 b1=bi;
 h1=[-1 1];
 sig1=h1*h1'+.01;
 p1=0.25;

 b2=bi;
 h2=[1 -1];
 sig2=h2*h2'+.01;
 p2=0.25;

 b3=bi;
 h3=[1 1];
 sig3=h3*h3'+.01;
 p3=0.25;

 for j=n+1:m+n
   p00=p0;                       %FOR EACH FILTER
   p11=p1;
   p22=p2;
   p33=p3;

   s0=sum(b0.*h0');                                    %
   b0=b0+((1/sig0)*h0'*(r(j)-s0));                     %
   N0=1/(sqrt(2*pi*sig0))*exp((-(r(j)-s0).^2)/2*sig0); %  FOR EACH FILTER
   p0=N0*(p00+p11);                                    %
```

```
  s1=sum(b1.*h1');
  b1=b1+((1/sig1)*h1'*(r(j)-s1));
  N1=1/(sqrt(2*pi*sig1))*exp((-(r(j)-s1).^2)/2*sig1);
  p1=N1*(p22+p33);

  s2=sum(b2.*h2');
  b2=b2+((1/sig2)*h2'*(r(j)-s2));
  N2=1/(sqrt(2*pi*sig2))*exp((-(r(j)-s2).^2)/2*sig2);
  p2=N2*(p00+p11);

  s3=sum(b3.*h3');
  b3=b3+((1/sig3)*h3'*(r(j)-s3));
  N3=1/(sqrt(2*pi*sig3))*exp((-(r(j)-s3).^2)/2*sig3);
  p3=N3*(p22+p33);

  pn=p0+p1+p2+p3;        %NORMALIZER
  p0=p0/pn;              %FOR EACH FILTER
  p1=p1/pn;
  p2=p2/pn;
  p3=p3/pn;

  b0=b0*(p0/(p0+p1))+b1*(p1/(p0+p1)); %FOR EACH FILTER
  b1=b2*(p2/(p2+p3))+b3*(p3/(p2+p3));
  b2=b0;
  b3=b1;
 end

 b0'            %    FOR EACH FILTER
 b1'
 b2'
 b3'

 for k=n+1:m+n
%   z0(k)=(b0(1:n+1)'*r(k:-1:k-n)');  %  NONRECURSIVE CASE
%   z1(k)=(b1(1:n+1)'*r(k:-1:k-n)');  %
  z0(k)=(z0(k-1:-1:k-n)*(-b0(2:n+1))+r(k))/b0(1);
  z1(k)=(z1(k-1:-1:k-n)*(-b1(2:n+1))+r(k))/b1(1);
  s0(k)=(1-(z0(k)^2));    %^2;
  se0(k)=s0(k)/sqrt(sig0);
  s1(k)=(1-(z1(k)^2));    %^2;
  se1(k)=s1(k)/sqrt(sig1);
 end

 sec=input('   ENTER 1 for 1&3;2 for 2&4 channels.....:');
```

```
 while sec==1
%   plot(b0(1,:),b0(2,:),'wo'); %
%   plot(bi(1,:),bi(2,:),'wx')  %
%   axis([-2 2 -2 2]);          %
%   hold on                     %  TRAJECTORIES
%   grid;                       %
%   xlabel('b0');               %
%   ylabel('b1');               %
%   pause                       %
  %  plot(s1(n+1:n+m),'w')      %
  %  axis([0 500 -1 5]);        %
  %  grid;                      % RECOVERY ERRORS
  %  xlabel('Iterations');      %
  %  ylabel('(u(k)-u(k)) ');    %
  %  pause                      %
  [n2,x2]=hist(se0,30);         %
  k2=-4:.01:4;                  %
  gt=1/sqrt(2*pi)*exp((-(k2.^2))/2); %
  m2=max(gt);                   %
  [xb2,yb2]=bar(x2,((n2/max(n2))*m2)); %HISTOGRAMS
  plot(k2,gt,'w--',xb2,yb2,'w'); %
  ort=mean(se0);                %
  var=(std(se0)^2);             %
  gtext(['mean=',num2str(ort)]); %
  gtext(['var=',num2str(var)]); %
  pause                         %
 sec=input('    ENTER 1 for 1&3;2 for 2&4 channels.....:');
 end

 while sec==2                   % different parts are as shown above
%   plot(b1(1,:),b1(2,:),'wo');
%   plot(bi(1,:),bi(2,:),'wx')
%   axis([-2 2 -2 2]);
%   hold on
%   grid;
%   xlabel('b0');
%   ylabel('b1');
%   pause
   %    plot(s1(n+1:n+m),'w')
   %    axis([0 500 -1 5]);
   %    grid;
   %    xlabel('Iterations');
   %    ylabel('(u(k)-u(k)) ');
   %    pause
  [n2,x2]=hist(se1,30);
  k2=-4:.01:4;
  gt=1/sqrt(2*pi)*exp((-(k2.^2))/2);
```

```
  m2=max(gt);
  [xb2,yb2]=bar(x2,((n2/max(n2))*m2));
  plot(k2,gt,'w--',xb2,yb2,'w');
  ort=mean(se1);
  var=(std(se1)^2);
  gtext(['mean=',num2str(ort)]);
  gtext(['var=',num2str(var)]);%
  pause
 sec=input('    ENTER 1 for 1&3;2 for 2&4 channels.....:');
 end

 hold on
 sel=input('    ENTER 1 to continue.....:');
end

end;
```

C. BLIND EQUALIZER FOR LINEAR, TIME-INVARIANT, STABLE CHANNELS BY USING KALMAN FILTER AND HMM ALGORITHM

```
% Filename : trynew.m
% Title    : Blind equalizer for linear,time-invariant,stable channels
%            by using Kalman filtering and HMM algorithm
% Date of last revision : 25 Aug 1993
% Comments : This program produces an array of random digital NRZ, BPSK
%            signal values, (either + or -1) to represent the message
%            signal.The length of message signal depends on the sampling
%            amount m(of course bigger m makes better estimation).Also
%            applies a FEC code by HMM rate choosen by the user.Then
%            this signal is applied to a channel such as:[x(t) x(t-1) ...
%            x(t-n)] where the coefficiants defined by user.To estimate
%            the equalizer coefficiants, by Kalman filtering and HMM
%            approach.Overall impulse response of the system, sent and
%            recived sequences are plotted as well as the errors due to
%            the equalizer
% Input variables :
%          m : Length of the message
% Output variables :None
% Associated functions : bpskgen.m
%                        conenc.m
%                        awgn.m
%                        eqcondec.m

m=input('ENTER THE LENGTH OF MESSAGE YOU WANT:');
mes=bpskgen(m);
```

```
sig=conenc(mes);
xn=zeros(size(sig));
num=[1,0.6,0.4,0.3]; den=zeros(size(num)); den(1)=1;
% num=[1,0]; den=[1,-0.6];
xt=filter(num,den,sig);     %  channel
%xn=xt;                     %  no noise
xn=awgn(xt,0.277);          %  13 dB SNR
%xn=awgn(xt,0.447);         %  5 dB SNR
%xn=awgn(xt,0.577);         %  3 dB SNR
%xn=awgn(xt,1);             %  0 dB SNR
h=dimpulse(num,den,20);
[info,tmax]=eqcondec(xn,h);
pause
subplot(223),plot(mes(1:tmax),'w'), title('actual message')
subplot(224),plot(mes(1:tmax)-info(1:tmax),'w'), title('errors')
end  % of program


function Y = awgn(X,sigma)
%           AWGN GENERATOR
%           07-31-1993
% Awgn is an m-file that adds awgn noise to the matrix X
% Where sigma is standart deviation of the noise
% Ec/No=1/(2*sigma^2).
[rr,cc]=size(X);
W=randn(rr,cc)+i*randn(rr,cc);
Y=X+sigma.*W;
disp('AWGN IS ADDED TO SIGNAL');


function u=bpskgen(k)
%               BPSK GENERATOR
%               08-01-1993
% This m-file accepts k the number of bits that will be returned
% in the vector u which is a BPSK sequence of {+1,-1}
u=sign(randn(size(1:k)));
disp('A random message is generated and coded in bipolar NRZ form');


function s=conenc(u)
%               CONVOLUTIONAL ENCODER
%               07-31-1993
% This m-file is a feedforward convolutional encoder for state transition
% matrix F,output sequence matrix G and input message matrix u. The number
% of convolution schemes can be increased by adding F & G matrices for new
% rates.
```

```
disp('To use rate=1/2 convolutional encoder ENTER 1');
disp('To use rate=1/3 convolutional encoder ENTER 2');
disp('To quit encoding                    ENTER 0');
sel=input('')
while sel==1,
  F=[1,2;
    3,4;
    1,2;
    3,4];
  G=[0,-1,1,-1, 1,1,-1, 1,-1;
    0,-1,1, 1,-1,1,-1,-1, 1];

  x=1;
  s=zeros(size(G(:,1)'));
  for t=1:length(u)
     x=F(round(x),round(1.5+u(t)/2));
     if x==1,
       sf=G(:,(round(x)+round(1.5+u(t)/2)));
       elseif x==2,
         sf=G(:,(round(x)+1+round(1.5+u(t)/2)));
         elseif x==3,
           sf=G(:,(round(x)+2+round(1.5+u(t)/2)));
           elseif x==4,
             sf=G(:,(round(x)+3+round(1.5+u(t)/2)));
     end  % for if statement
     s=[s,sf'];
  end   % for for loop
disp('Message is encoded');
disp('To quit encoding ENTER 0');
sel=input('')
end    % for while loop
while sel==2,
  F=[1,5;
    1,5;
    2,6;
    2,6;
    3,7;
    3,7;
    4,8;
    4,8];
  G=[0,-1,1,-1, 1,-1, 1, 1,-1, 1,-1, 1,-1;
    0,-1,1,-1,-1, 1,-1, 1,-1, 1, 1,-1, 1;
    0,-1,1,-1, 1,-1, 1,-1, 1,-1, 1,-1, 1];

  x=1;
  s=zeros(size(G(:,1)'));
  for t=1:length(u)
```

```
      x=F(round(x),round(1.5+u(t)/2));
      if x==1|2,
        sf=G(:,(round(x)+round(1.5+u(t)/2)));
        elseif x==3|4,
          sf=G(:,(round(x)+1+round(1.5+u(t)/2)));
          elseif x==5|6,
            sf=G(:,(round(x)+2+round(1.5+u(t)/2)));
            elseif x==7|8,
              sf=G(:,(round(x)+3+round(1.5+u(t)/2)));
      end   % for if statement
      s=[s,sf'];
  end    % for for loop
disp('Message is encoded');
disp('To quit encoding ENTER 0');
sel=input('')
end    % for while loop


function [uh,tmax]=eqcondec(x,h)
%            EQUALIZER AND CONVOLUTIONAL DECODER
%                 07-30-1993
% This m-file is a channel equalizer and convolutional decoder for state
% transition matrix F,output sequence matrix G ,channel parameter matrix
% x and input signal s.
% Simply uses the Kalman filtering algorithm with Hidden Markov Models.
% The number of convolution schemes can be increased by adding F & G matrices
% for new rates
disp('To use rate=1/2 convolutional decoder ENTER 1');
disp('To use rate=1/3 convolutional decoder ENTER 2');
sel=input('')
if sel==1,
  F=[1,2;
     3,4;
     1,2;
     3,4];
  G=[0,-1,1,-1, 1,1,-1, 1,-1;
     0,-1,1, 1,-1,1,-1,-1, 1];
end    % for if statement
if sel==2,
  F=[1,5;
     1,5;
     2,6;
     2,6;
     3,7;
     3,7;
     4,8;
     4,8];
```

```
  G=[0,-1,1,-1, 1,-1, 1, 1,-1, 1,-1, 1,-1;
     0,-1,1,-1,-1, 1,-1, 1,-1, 1, 1,-1, 1;
     0,-1,1,-1, 1,-1, 1,-1, 1,-1, 1,-1, 1];
end    % for if statement

nf=length(G(:,1));
ns=length(F(:,1));
n=15;        % order of the filter
th=zeros(n,ns);
th(1,:)=5.0*ones(1,ns);
e2=zeros(ns,1);

P=1*eye(n);
tmin=round(n/nf)+1;
tmax=floor(100/nf)-1;
point=zeros(tmax+1,ns);
uhat=zeros(tmax+1,ns);
for tk=tmin:tmax,
   phi=toeplitz(x(tk*nf+1:tk*nf+nf),x(tk*nf+1:-1:tk*nf-n+2))';
   fact1=P*phi;
   fact2=fact1'*phi;
   fact=inv(eye(length(fact2))+fact2);
   K=fact1*fact;
   en2=inf*ones(ns,1);          % start error for new layer at infinity
   if ns==4,
    for j=1:ns  % states
      for i=1:2  % inputs
      m=F(j,i);
      if m==1,                                          %
        v=G(:,(m+i))-phi'*th(:,j);                       %
        elseif m==2,                                    %
          v=G(:,(m+1+i))-phi'*th(:,j);                    % RATE 1/2
            elseif m==3,                                %
              v=G(:,(m+2+i))-phi'*th(:,j);               %
                elseif m==4,                            %
                  v=G(:,(m+3+i))-phi'*th(:,j);           %
      end                                               %
      etemp=e2(j)+v'*fact*v;
      if etemp<en2(m),
        en2(m)=etemp;
        thn(:,m)=th(:,j)+K*v;
        point(tk+1,m)=j;
        uhat(tk+1,m)=i;
      end    % for if statement
      end     % for loop of i
    end     % for loop of j
   end      % for while loop
```

```
    if ns==8,
     for j=1:ns  % states
      for i=1:2  % inputs
      m=F(j,i);
      if m==1|2,                                    %
        v=G(:,(m+i))-phi'*th(:,j);                  %
        elseif m==3|4,                              %
         v=G(:,(m+1+i))-phi'*th(:,j);               %
          elseif m==5|6,                            %  RATE 1/3
           v=G(:,(m+2+i))-phi'*th(:,j);             %
            elseif m==7|8,                          %
             v=G(:,(m+3+i))-phi'*th(:,j);           %
      end                                           %
            etemp=e2(j)+v'*fact*v;
      if etemp<en2(m),
        en2(m)=etemp;
        thn(:,m)=th(:,j)+K*v;
        point(tk+1,m)=j;
        uhat(tk+1,m)=i;
      end    % for if statement
      end     % for loop of i
     end     % for loop of j
    end      % for while loop
    th=thn;
    e2=en2;
    P=P-fact1*fact*fact1';
end
[em,m]=min(e2);
thf=th(:,m);
% estimated message
while m ~= 0,
  uh(tk)=2*uhat(tk+1,m)-3;
  m=point(tk+1,m);
  tk=tk-1;
end
disp('Signal is decoded, now look to the graphs');
pause

clg;
hold off
c=conv(h,thf);
subplot(221), plot(c,'ow'),title('impulse resp of ch+eq')
%nu=min([length(u), length(uh)]);
subplot(222),plot(uh(1:tmax),'w'), title('estimated message')
end  % of program
```

D. SIMULATION OF THE HMM AND KALMAN FILTERING ALGORITHM IN FLAT RAYLEIGH FADING CHANNEL

```
% Filename : simul.m
% Title    : Simulation of combined equalizer and FEC decoder (HMM and
%            Kalman filtering algorithm) in flat Rayleigh fading channel
% Date of last revision : 15 Sep 1993
% Comments : This program simulates the combined equalizer and FEC decoder
%            (HMM and Kalman filtering algorithm) in flat Rayleigh fading
%            channel. Works similar to previuos programs, uses DBPSK
%            instead of BPSK.Industrial standart, code rate 1/2 & constraint
%            length 7 FEC encoder is employed
% Input variables :
%          m : Length of the message sequence
% Output variables :None
% Associated functions : awgn.m
%                        bpskgen.m
%                        conenc64.m
%                        difecod.m
%                        pskmod.m
%                        fade.m
%                        pskdmod.m
%                        eqcod64.m

m=input('ENTER THE LENGTH OF MESSAGE YOU WANT:');
mes=bpskgen(m);
cme=conenc64(mes,F64,G64);
len=length(cme);
dcm=difecod(cme);
sig=pskmod(dcm,7);
sig1=sig';
sig2=sig1(:);
sig3=exp(i*pi/4)*sig2;
disp('POWER IS SPLITTED TO I & Q CHANNELS');
I=fade(0.01,(len+1)*7);
Q=fade(0.01,(len+1)*7);
sig4=I.*real(sig3)+j*(Q.*imag(sig3));
disp('Rayleigh fading is applied to signal');
sig5=awgn(sig4,0.277);
[cmm,a]=pskdmod(sig,7,len);
[info,tmax]=eqcod64(cmm,F64,G64);
pause
subplot(211),plot(mes(1:tmax),'w'), title('actual message')
subplot(212),plot(info(1:tmax),'w'),title('received message')
pause
print -dmeta;
```

```
clg;
subplot(211),plot(mes(1:tmax)-info(1:tmax),'w'), title('errors')
end  % of program


function s=conenc64(u,F64,G64)
%                CONVOLUTIONAL ENCODER
%                08-30-1993
% This m-file is a feedforward industrial standart convolutional encoder
% for state transition matrix F64,output sequence matrix G64 and input
% message matrix u. F64 & G64 matrices are given as a file named incod64.mat
% Be sure to load that file before working with this function
x=1;
s=zeros(size(G64(:,1)'));
for t=1:length(u)
  x=F64(round(x),round(1.5+u(t)/2));
  if x==1|9,
    sf=G64(:,(0+round(1.5+u(t)/2)));
    elseif x==2|10,
      sf=G64(:,(2+round(1.5+u(t)/2)));
      elseif x==3|11,
        sf=G64(:,(4+round(1.5+u(t)/2)));
        elseif x==4|12,
          sf=G64(:,(6+round(1.5+u(t)/2)));
          elseif x==5|13,
            sf=G64(:,(8+round(1.5+u(t)/2)));
            elseif x==6|14,
              sf=G64(:,(10+round(1.5+u(t)/2)));
              elseif x==7|15,
                sf=G64(:,(12+round(1.5+u(t)/2)));
                elseif x==8|16,
                  sf=G64(:,(14+round(1.5+u(t)/2)));
                  elseif x==17|25,
                    sf=G64(:,(16+round(1.5+u(t)/2)));
                    elseif x==18|26,
                      sf=G64(:,(18+round(1.5+u(t)/2)));
                      elseif x==19|27,
                        sf=G64(:,(20+round(1.5+u(t)/2)));
                        elseif x==20|28,
                          sf=G64(:,(22+round(1.5+u(t)/2)));
                          elseif x==21|29,
                            sf=G64(:,(24+round(1.5+u(t)/2)));
                            elseif x==22|30,
                              sf=G64(:,(26+round(1.5+u(t)/2)));
                              elseif x==23|31,
                                sf=G64(:,(28+round(1.5+u(t)/2)));
                                elseif x==24|32,
```

```
                    sf=G64(:,(30+round(1.5+u(t)/2)));
                    elseif x==33|41,
                      sf=G64(:,(32+round(1.5+u(t)/2)));
                      elseif x==34|42,
                        sf=G64(:,(34+round(1.5+u(t)/2)));
                        elseif x==35|43,
                          sf=G64(:,(36+round(1.5+u(t)/2)));
                          elseif x==36|44,
                            sf=G64(:,(38+round(1.5+u(t)/2)));
                          elseif x==37|45,
                          sf=G64(:,(40+round(1.5+u(t)/2)));
                        elseif x==38|46,
                        sf=G64(:,(42+round(1.5+u(t)/2)));
                      elseif x==39|47,
                      sf=G64(:,(44+round(1.5+u(t)/2)));
                    elseif x==40|48,
                    sf=G64(:,(46+round(1.5+u(t)/2)));
                  elseif x==49|57,
                  sf=G64(:,(48+round(1.5+u(t)/2)));
                elseif x==50|58,
                sf=G64(:,(50+round(1.5+u(t)/2)));
              elseif x==51|59,
              sf=G64(:,(52+round(1.5+u(t)/2)));
            elseif x==52|60,
            sf=G64(:,(54+round(1.5+u(t)/2)));
          elseif x==53|61,
          sf=G64(:,(56+round(1.5+u(t)/2)));
        elseif x==54|62,
        sf=G64(:,(58+round(1.5+u(t)/2)));
      elseif x==55|63,
      sf=G64(:,(60+round(1.5+u(t)/2)));
    elseif x==56|64,
    sf=G64(:,(62+round(1.5+u(t)/2)));
  end   % for if statement
  s=[s,sf'];
end    % for for loop
disp('FEC is applied to Message(ENCODED)');


function [uh,tmax]=eqcod64(x,F64,G64)
%            EQUALIZER AND CONVOLUTIONAL DECODER
%                 08-21-1993
% This m-file is a channel equalizer and convolutional decoder for state
% transition matrix F64,output sequence matrix G64,channel parameter matrix
% x and input signal s.
% Industrial standart Rate 1/2 code with constraint length 7 is used F64 &
% G64 matrices are defined in incod64.mat. Be sure it is loaded.
```

```
% Simply uses the Kalman filtering algorithm with Hidden Markov Models.
% The number of convolution schemes can be increased by adding F & G matrices
% for new rates
nf=length(G64(:,1));
ns=length(F64(:,1));
n=15;       % order of the filter
th=zeros(n,ns);
th(1,:)=5.0*ones(1,ns);
e2=zeros(ns,1);
P=1*eye(n);
tmin=round(n/nf)+1;
tmax=floor(100/nf)-1;
point=zeros(tmax+1,ns);
uhat=zeros(tmax+1,ns);
for tk=tmin:tmax,
   phi=toeplitz(x(tk*nf+1:tk*nf+nf),x(tk*nf+1:-1:tk*nf-n+2))';
   fact1=P*phi;
   fact2=fact1'*phi;
   fact=inv(eye(length(fact2))+fact2);
   K=fact1*fact;
   en2=inf*ones(ns,1);          % start error for new layer at infinity
   for j=1:ns  % states
      for i=1:2  % inputs
      m=F64(j,i);
      if m==1|9,
        v=G64(:,(0+i))-phi'*th(:,j);
        elseif m==2|10,
          v=G64(:,(2+i))-phi'*th(:,j);
          elseif m==3|11,
            v=G64(:,(4+i))-phi'*th(:,j);
            elseif m==4|12,
              v=G64(:,(6+i))-phi'*th(:,j);
              elseif m==5|13,
                v=G64(:,(8+i))-phi'*th(:,j);
                elseif m==6|14,
                  v=G64(:,(10+i))-phi'*th(:,j);
                  elseif m==7|15,
                    v=G64(:,(12+i))-phi'*th(:,j);
                    elseif m==8|16,
                      v=G64(:,(14+i))-phi'*th(:,j);
                      elseif m==17|25,
                        v=G64(:,(16+i))-phi'*th(:,j);
                        elseif m==18|26,
                          v=G64(:,(18+i))-phi'*th(:,j);
                          elseif m==19|27,
                            v=G64(:,(20+i))-phi'*th(:,j);
                            elseif m==20|28,
```

```
                v=G64(:,(22+i))-phi'*th(:,j);
                elseif m==21|29,
                  v=G64(:,(24+i))-phi'*th(:,j);
                  elseif m==22|30,
                    v=G64(:,(26+i))-phi'*th(:,j);
                    elseif m==23|31,
                      v=G64(:,(28+i))-phi'*th(:,j);
                      elseif m==24|32,
                        v=G64(:,(30+i))-phi'*th(:,j);
                        elseif m==33|41,
                          v=G64(:,(32+i))-phi'*th(:,j);
                          elseif m==34|42,
                            v=G64(:,(34+i))-phi'*th(:,j);
                            elseif m==35|43,
                              v=G64(:,(36+i))-phi'*th(:,j);
                              elseif m==36|44,
                                v=G64(:,(38+i))-phi'*th(:,j);
                                elseif m==37|45,
                                  v=G64(:,(40+i))-phi'*th(:,j);
                                elseif m==38|46,
                                v=G64(:,(42+i))-phi'*th(:,j);
                              elseif m==39|47,
                              v=G64(:,(44+i))-phi'*th(:,j);
                            elseif m==40|48,
                            v=G64(:,(46+i))-phi'*th(:,j);
                          elseif m==49|57,
                          v=G64(:,(48+i))-phi'*th(:,j);
                        elseif m==50|58,
                        v=G64(:,(50+i))-phi'*th(:,j);
                      elseif m==51|59,
                      v=G64(:,(52+i))-phi'*th(:,j);
                    elseif m==52|60,
                    v=G64(:,(54+i))-phi'*th(:,j);
                  elseif m==53|61,
                  v=G64(:,(56+i))-phi'*th(:,j);
                elseif m==54|62,
                v=G64(:,(58+i))-phi'*th(:,j);
              elseif m==55|63,
              v=G64(:,(60+i))-phi'*th(:,j);
            elseif m==56|64,
            v=G64(:,(62+i))-phi'*th(:,j);
end
      etemp=e2(j)+v'*fact*v;
if etemp<en2(m),
  en2(m)=etemp;
  thn(:,m)=th(:,j)+K*v;
  point(tk+1,m)=j;
```

```
        uhat(tk+1,m)=i;
      end    % for if statement
      end     % for loop of i
    end     % for loop of j
  th=thn;
  e2=en2;
  P=P-fact1*fact*fact1';
end
[em,m]=min(e2);
thf=th(:,m);
% estimated message
while m ~= 0,
 uh(tk)=2*uhat(tk+1,m)-3;
 m=point(tk+1,m);
 tk=tk-1;
end
disp('Signal is decoded');


function out=pskmod(in,N)
%            PSK MODULATOR
%            09-03-1993
% This M file accepts the differentially encoded vector in
% and # of samples per symbol N.  Then generates a matrix
% with dimensions (length(in)xN)
out=in'*ones(1,N);
disp('MESSAGE DPSK MODULATED');


function [out,a]=pskdmod(sig,N,m)
%            PSK DEMODULATOR
%            09-02-1993
% This function accepts fading and AWGN effected sig,
% # of samples per symbol N and total # of symbols m
% Then for each symbol takes the difference and sum of
% present and past symbol sums and squares their absolute
% values. If difference<sum decides input bit is 0(rep. -1)
o=ones(1,m);
for i=2:m+1,
  dif(i-1)=abs(sum(sig(i,:))-sum(sig(i-1,:))).^2;
  su(i-1)=abs(sum(sig(i,:))+sum(sig(i-1,:))).^2;
  met(i-1)=dif(i-1)-su(i-1);
  if met(i-1)<0;
    o(i-1)=-1;
  end
end
a=met;
```

```
out=o;
disp('SIGNAL IS DEMODULATED');


function difout=difecod(n)
%           DIFFERENTIAL ENCODER
%           08-31-1993
% This M file accepts an input vector n(which is bipolar NRZ
% coded form of output sequence) and differentially encodes it
% with a reference bit 1(which will be inserted at the beginning
% of the vector.
y=[1,zeros(1,length(n))];
for i=1:length(n),
   if n(i)==-1,
      n(i)=0;
   end
   y(i+1)=xor(n(i),y(i));
end
for j=1:length(n),
   if n(i)==0,
      n(i)=-1;
   end
   if y(i)==0,
      y(i)=-1;
   end
end
difout=y;
disp('MESSAGE DIFFERENTIALLY ENCODED');


function out=fade(dps,n)
%           FADING GENERATOR
%           09-04-1993
% This function accepts differential phase shift dps and
% # of samples n. Then creates n normal distributed
% variables passes these through two RC filters
s=exp(-dps/2.146193);
ss=((1-s^2)^3/(1+s^2))^.25;
b=[ss];
a=[1 -s];
t=randn(n,1);
k=filter(b,a,t);
out=filter(b,a,k);
```

LIST OF REFERENCES

1. S.Haykin, *Adaptive Filter Theory*, Prentice-Hall, New York, NY, 1991.

2. O.Shalvi and E.Weinstein, "New Criteria for Blind Deconvolution of Nonminimum Phase Systems," *IEEE Transactions on Information Theory*, Vol. 36, pp.312-321, March 1990.

3. C.R.Johnson,Jr., "Admissibility in Blind Adaptive Channel Equalization," *IEEE Control Systems Mag., Special Issue on Signal Processing*, Vol 11, pp.3-15, January 1991.

4. R.A.Iltis, J.J.Shynk, and G.Krishnamurty, "Bayesian Algorithms for Blind Equalization using Parallel Adaptive Filtering," Center for Information Processing Research, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, May 1991.

5. L.Rabiner and B.H.Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

6. B.Kahraman, "Performance Evaluation of UHF Fading Satellite Channel by Simulation for Different Modulation Schemes," Master's Thesis, Naval Postgraduate School, Monterey, CA, December 1992.

7. W.A.Sethares, R.A.Kennedy, and Zhenguo Gu, "An Approach to Blind Equalization of Non-minimum Phase Systems," *IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-92*, pp.1529-1532, Toronto, Canada, 1991.

8. Zhi Ding, C.R.Johnson,Jr., and R.A.Kennedy, "Local Convergence of Globally Convergent Blind Adaptive Equalization Algorithms," *IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-92*, pp.1533-1536, 1991.

9. J.R.Treichler, V.Wolff, and C.R.Johnson,Jr., "Observed Misconvergence in the Constant Modulus Adaptive Algorithm," *IEEE Proceedings of 25th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 1991.

10. Zhi Ding, R.A.Kennedy, and K.Yamazaki, "Globally Convergent Blind Equalization Algorithms For Complex Data Systems," *IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-92*, pp.IV-553-IV-556, March 1992.

11. M.V.Eyuboglu and S.U.H.Qureshi, "Reduced-State Sequence Estimation with Set Partitioning and Decision Feedback," *IEEE Transactions on Communications*, Vol.36, pp.13-20, January 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library Code 52 Naval Postgraduate School Monterey, CA 93943-5000	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor R. Cristi, Code EC/Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Professor P. Pace, Code EC/Pc Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Deniz Kuvvetleri Komutanlığı Personel Daire Başkanlığı Bakanlıklar, Ankara, TURKEY	1
7.	Deniz Harp Okulu Komutanlığı Tuzla, İstanbul, TURKEY	2
8.	Gölcük Tersanesi Komutanlığı Gölcük, Kocaeli, TURKEY	2
9.	Taşkızak Tersanesi Komutanlığı Taşkızak, İstanbul, TURKEY	2
10.	Mehmet Kutlu 9-10. Kısım B-31 Blok Daire:34 Ataköy, İstanbul, TURKEY	1